Хмельницький національний університет
Факультет програмування та
комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерної інженерії та системного програмування

# ДИПЛОМНА РОБОТА МАГІСТРА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Розподілена система моделювання мурашиного алгоритму в корпоративних комп'ютерних мережах»

ДРКІСПр. 015099.19.02.13 ПЗ

Виконав: студент 2 курсу, група КІ2м-19-1 _____ Овсяк.О.В.
Підпис Ініціали, прізвище

Керівник канд. техн. наук, доцент _____ Медзатий Д.М.
Науковий ступінь, вчене звання Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КІСП, д.т.н., проф.
Т.О. Говорущенко
__ _____ 2021 р.

Хмельницький, 2021

# ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем

Кафедра Комп'ютерної інженерії та системного програмування

Освітній рівень магістр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма освітньо-наукова програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Зав. кафедри Р.О.Говорущенко

"07" 09 2020р.

## ЗАВДАННЯ
## НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Овсяк Олександр Валентинович

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Розподілена система моделювання мурашиного алгоритму в корпоративних комп'ютерних мережах

Керівник проекту (роботи) Медзатий Д.М., к.т.н., доцент

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.01.2021 р. № 7

2. Строк подання студентом проекту (роботи) на кафедру 31.05.2021 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Структурно складові корпоративної мережі

Моделювання та проектування компютерної мережі

Оптимізація мурашиними колоніями

Ефективність розподілених систем

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів дипломного проекту (роботи)

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |

| | | | |
|---|---|---|---|
| Нормоконтроль | Лисенко С.М., професор кафедри КІСП | | |
| Антиплагіат | Нічепорук А.О., доцент кафедри КІСП | | |

7. Дата видачі завдання «_01»_____09_____2020р.

## КАЛЕНДАРНИЙ ПЛАН

| №з/п | Назва етапів (розділів) дипломного проекту (роботи) | Термін виконання етапів проекту (роботи) | Примітка |
|---|---|---|---|
| 1 | Вибір напряму дослідження та узгодження тематики ДРМ з керівником | 01.09.2020 | виконано |
| 2 | Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження | 05.10.2020 | виконано |
| 3 | Робота над розділом 1 – огляд концепції мурашиних алгоритмів. Аналіз відомих рішень та засобів | 05.11.2020 | виконано |
| 4 | Робота над розділом 2 – розробка моделі розподіленої системи | 07.12.2020 | виконано |
| 5 | Робота над науковою статтею | 05.01.2021 | виконано |
| 6 | Робота над розділом 3 – розробка методу оптимізації мережі мурашиними алгоритмами шляхом балансування мережі | 01.02.2021 | виконано |
| 7 | Робота над розділом 4 – розробка і постановка експерименту із мурашиними алгоритмами | 11.03.2021 | виконано |
| 8 | Оформлення пояснювальної записки згідно вимог | 15.04.2021 | виконано |
| 9 | Попередній захист ДРМ | 21.04.2021 | виконано |
| 10 | Захист ДРМ на засіданні ЕК | 31.05.2021 | |

**Студент** _____ Овсяк О.В.
Підпис Ініціали, прізвище

**Керівник проекту (роботи)** _____ Медзатий Д.М.
Підпис Ініціали, прізвище

# РЕФЕРАТ

Тема дипломної роботи: «Розподілена система моделювання мурашиного алгоритму в корпоративних комп’ютерних мережах»

Автор роботи: Овсяк Олександр Валентинович

Керівник роботи: Медзатий Д.М.

Пояснювальна записка: 123 с., 12 рис., 1 табл., 3 дод., 61 джерел.

ПЕРЕЛІК КЛЮЧОВИХ СЛІВ (6-8) ЧЕРЕЗ КОМУ: розподілена система, комп'ютерна мережа, мурашиний алгоритм, оптимізація, мурашина колонія.

Об’єктом дослідження є процес оптимізації навантаження розподілених систем.

Предметом дослідження є розподілена система моделювання мурашиного алгоритму в корпоративних комп’ютерних мережах.

Метою дипломної роботи є оптимізація шляхом досягнення результатів балансування навантаження, іншими словами, мінімізація затримки та часу відгуку при одночасному збільшенні пропускної здатності простими словами поліпшити продуктивність PC.

Для розв’язання поставлених задач використовувалися методи теорії розподілених систем, теорії компютерних мереж, метод поведінки колонії біологічних мурах, мурашиних алгоритмів.

Наукова новизна отриманих результатів: запропоновано удосконалений метод для балансування навантаження в розподілених системах на основі численних колоній мурашок була запропонована оптимізація. Використання кількох гнізд, або колоній мурашок у процесі пошуку, допомогло підвищити швидкість обміну інформацією по всіх вузлах системи. Крім того, динамічний обмін інформацією та її повне розповсюдження - це інші основні характеристики, що відрізняють цей підхід. Результати показали ефективність запропонованої моделі в порівнянні зі стандартним алгоритмам викрадення роботи з точки зору

кількості зайнятих вузлів та витраченого часу для досягнення ефективності понад 15 %

Практична значимість в отриманих результатів полягає у результаті виконання наукового дослідження розроблений алгоритм оптимізації навантаження на розподілену систему що в подальшому дає широкий спектр можливостей з оптимізацією мереж для покращення ефективності.

За темою дипломної роботи опубліковано матеріали у XII ВСЕУКРАЇНСЬКІЙ НАУКОВО-ПРАКТИЧНІЙ КОНФЕРЕНЦІЇ АПКН 2020

# ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ 6
ВСТУП 7
1 СТРУКТУРНО СКЛАДОВІ КОРПОРАТИВНОЇ МЕРЕЖІ 10
1.1 Огляд та поняття корпоративних мереж 10
1.2 Необхідне обладнання для побудови комп'ютерної корпоративної мережі
1.3 Переваги і можливості корпоративної мережі 12
1.4 Складнощі при створенні корпоративної мережі 14
1.5 Аналіз відомих методів і засобів 15
1.6 Кластерні обчислення 17
1.7 Сіткові обчислення 19
1.8 Хмарні обчислення 21
1.9 Організація розподіленої інформаційні системи 23
1.10 Інтеграція корпоративних додатків 24
1.11 Передача файлів 26
1.12 Спільна база даних 27
1.13 Обмін повідомленнями 27
1.14 Поширені системи 28
1.15 Системи виявлення вторгнень 29
1.16 Висновки 30
2 РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ КОРПОРАТИВНОЇ МЕРЕЖІ 31
2.1 Вибір і аналіз комутаційного обладнання 33
2.2 Вибір серверної інфраструктури мережі 39
2.3 Вибір засобів моніторингу мережі 44
2.4 Розробка плану IP-адресації 45
2.5 Створення схеми комп'ютерної мережі 46
2.6 Висновки 47
3 МУРАШИНИЙ АЛГОРИТМ 48
3.1 Оптимізація колонії мурашок 49

3.2 Особливості мурашиних систем 51
3.3 Система колоній мурашок 53
3.4 Феромон мурашиного алгоритму 53
3.5 Гібридизація та підвищення продуктивності 55
3.6 ANTS 56
3.7 Механізми оновлення маршруту 56
3.8 Проблеми оптимізації 58
3.9 Квадратичне завдання на присвоєння 59
3.10 Застосування підходів ОПМ до проблем 60
3.11 Збіжності алгоритмів мурашиних колоній 62
3.12 Висновки 63
4 ЕФЕКТИВНІСТЬ РОЗПОДІЛЕНИХ СИСТЕМ 64
4.1 Постановка експерименту 66
4.2 Аналіз розподіленої системи балансування 67
4.3 Інтерфейс програмного забезпечення виявлення кібер-загроз 72
4.4 Обмін інформацією при оптимізації колонії мурашок 73
4.5 Запропонована стратегія 74
4.6 Результати 76
ВИСНОВКИ 82
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 84
ДОДАТОК А (Лістинг програмного забезпечення оптимізації корпоративної мережі мурашиним алгоритмом) 90
ДОДАТОК Б (Статі за результатами дослідження) 93
ДОДАТОК В (Презентація доповіді) 106

# СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

МС - Мурашина система

ОПМ - Оптимізація колонії мурашок

СУБД - Система управління базами даних

ККМ - Комп'ютерна корпоративна мережа

IDS - Системи виявлення вторгнень

ПЗ - Програмне забезпечення

ОС - Операційна система

БД - База даних

# ВСТУП

Проблеми оптимізації дуже важливі як у науковій, так і в промисловій галузі. Деякі реальні приклади цих проблем оптимізації - це графік розкладу руху, графік розподілу часу для медсестер, планування руху поїздів, планування пропускної спроможності, проблеми продавців, проблеми з маршрутом руху автомобілів, проблема планування роботи в групових магазинах, оптимізація портфеля тощо. Розроблено багато алгоритмів оптимізації для з цієї причини. Оптимізація колонії мурашок - одна з них. Оптимізація колонії мурашок - це імовірнісний прийом для пошуку оптимальних шляхів. В інформатиці та дослідженнях алгоритм оптимізації колонії мурашок використовується для вирішення різних обчислювальних задач.

Актуальність роботи зумовлена тим, що навіть при наявності великої кількості вже реалізованих мурашиних алгоритмів необхідно і далі реалізовувати, і покращувати дані алгоритми для пошуку найкращого результату. Як мені відомо у літературі було знайдено багато інформації по балансуванні навантаження використовуючи ACO, проте мало робіт проводиться із балансуванням навантаження із розподіленими системами з ACO. У моїй роботі було показаний експеримент з ACO для балансування навантаження в розподілених системах.Цей експеримент повністю розподілений, в якому інформація динамічно оновлюється при кожному русі мурахи. Буде прийнята парадигма кількох колоній, така що кожен вузол буде посилати кольорову колонію по всій мережі.

Метою дипломної роботи є досягнення результатів балансування навантаження, іншими словами, мінімізація затримки та часу відгуку при одночасному збільшенні пропускної здатності простими словами поліпшити продуктивність PC.

Поставлена мета досягається розв’язанням таких основних задач: Щоб отримати максимальну продуктивність потрібно провести балансування навантаження для цього потрібно призначити системі однаковий обсяг робіт. Оскільки різні завдання не обов’язково вимагають однакової кількості обчислень,

роботи чи часу, балансування навантаження не означає просто прохання кожної машини виконати іменну кількість завдань. Натомість існує два різні підходи до збалансування навантаження обчислювальної системи. Якщо різні розміри завдань відомі заздалегідь, навантаження можна статично збалансувати під час компіляції; якщо, однак, різний розмір завдання невідомий, завдання призначаються різним процесорам динамічно під час виконання.

Об’єктом дослідження є процес оптимізації навантаження розподілених систем.

Предметом дослідження є розподілена система моделювання мурашиного алгоритму в корпоративних комп’ютерних мережах.

Наукова новизна отриманих результатів:

1) розроблено удосконалений метод для балансування навантаження в розподілених системах на основі численних колоній мурашок була запропонована оптимізація. Використання кількох гнізд, або колоній мурашок у процесі пошуку, допомогло підвищити швидкість обміну інформацією по всіх вузлах системи. Крім того, динамічний обмін інформацією та її повне розповсюдження - це інші основні характеристики, що відрізняють цей підхід.

Практична значимість в отриманих результатів полягає у результаті виконання наукового дослідження розроблений алгоритм оптимізації навантаження на розподілену систему що в подальшому дає широкий спектр можливостей з оптимізацією мереж для покращення ефективності. Результати показали ефективність запропонованої моделі в порівнянні зі стандартним алгоритмам викрадення роботи з точки зору кількості зайнятих вузлів та витраченого часу для досягнення ефективності понад 50%.

За темою дипломної роботи опубліковано матеріали у XII ВСЕУКРАЇНСЬКІЙ НАУКОВО-ПРАКТИЧНІЙ КОНФЕРЕНЦІЇ АПКН 202

# 1 СТРУКТУРНО СКЛАДОВІ КОРПОРАТИВНОЇ МЕРЕЖІ

## 1.1 Огляд та поняття корпоративних мереж

Згідно виконуваного завдання є необхідне залучення корпоративної мережі яка експлуатується в межах підприємства і представляє собою складну інфраструктуру, яка призначена для передачі великого обсягу різнорідних інформаційних потоків (телефонія, обмін даними, доступ в Інтернет, відеоконференції і т. Д.) В межах одного підприємства називається корпоративна мережа

Для побудови корпоративної мережі в межах магістерської роботи потрібні такі основні мережеві компоненти:

1) маршрутизатори;
2) мережеві кабелі;
3) комутатори;
4) концентратори;
5) бездротові точки доступу;
6) сервери;
7) мережеві інтерфейсні карти;

Маршрутизатором називається пристрій, який з'єднує дві або більше мереж або під мереж з комутацією пакетів. Він виконує дві основні функції: управління трафіком між цими мережами шляхом переадресації пакетів даних на призначені їм IP-адреси та надання можливості кільком пристроям використовувати одне і те ж Інтернет-з'єднання.

Мережевий пристрій, який підключає пристрої в мережі, дозволяючи передавати дані всередині мережі називається Комутатор

Мережевий концентратор ще один мережевий пристрій, який з'єднує безліч периферійних пристроїв у мережі, змушуючи їх діяти як єдиний сегмент, проте, крім комутатора або маршрутизатора; концентратор транслює дані по кожному з'єднанню, замість того, щоб направляти їх на певний пристрій.

Точка бездротового доступу забезпечує бездротове з'єднання. Точка доступу виконує роль маршрутизатора, перенаправляющие дані з одного пристрою на інший за допомогою бездротових частот. Найбільш поширеним протоколом є IEEE 802.11 (Wi-Fi), який лежить в основі програм бездротових мереж.

Мережеві кабелі використовуються для фізичного підключення мережевого пристрою до іншого. Залежно від підключаються пристроїв, топології мережі та розміру мережі можуть використовуватися різні типи кабелів. Наприклад, у невеликій офісній мережі кабель Ethernet є основним мережевим кабелем, який використовується для підключення до пристроїв на невеликій відстані, тоді як волоконно-оптичні кабелі є кращими для передачі даних на більшій відстані, наприклад, підключення від постачальника послуг Інтернету клієнт.

Мережевий сервер - головний комп’ютер у мережі. Цей комп’ютер широко відомий як сховище даних та програм, яке існує в мережі.Сервер забезпечує функціональність (послуги) для всіх інших пристроїв у мережі, і ці пристрої відомі як "клієнти".

Для того, щоб пристрій зміг підключитися до мережі, йому потрібна мережева карта інтерфейсу (NIC). NIC, також відомий як мережевий адаптер, дозволяє пристрою надсилати та отримувати дані через мережу за допомогою кабелю Ethernet або бездротового з'єднання.Без встановленої NIC пристрій не може підключитися до мережі.

Це основні компоненти, необхідні мережі, щоб забезпечити надійне та безпечне IT-середовище

## 1.2 Необхідне обладнання для побудови комп'ютерної корпоративної мережі

1) Для реалізації роботи було використано таке обладнання:
2) коммутатор CoreBuilder 5000 Switch, 12-slot. 2шт;
3) 30 робочі станції(комп*ютери);
4) сервер Dell EMC R230 1 шт.с;

5) маршрутизатор Cisco RV340W Wireless-AC Dual WAN Gigabit VPN Router 1шт;

6) мережева карта Supermicro AOC-SGP-i2;

Головне завдання при побудові корпоративної мережі - оптимізація обробки і розподілу інформаційних потоків. Найпомітнішою тенденцією в області технології побудови мережі є об'єднання пакетного трафіку і мови в одному каналі зв'язку. Це розмиває кордони областей застосування телекомунікаційних та мережевих технологій: відбувається так звана конвергенція мереж. Подібна тенденція робить вибір базових технологій побудови мережі, протоколів обміну і обладнання вельми нетривіальним завданням.

Вибір концепції побудови конкретної корпоративної мережі визначається цілою низкою чинників: затребувані інформаційні послуги, обсяги переданого трафіку, існуюча інфраструктура і т. д. Але існують і загальні вимоги до корпоративних мереж. Мережі підприємств повинні бути побудовані на основі перевірених технологій, що володіють такими якостями, як масштабованість, гнучкість, мультисервісних, і найголовніше - надійність.

Мережа сучасного підприємства, як правило, повинна підтримувати ряд найбільш затребуваних для бізнесу додатків і керованих сервісів. В першу чергу це:

1) можливість високошвидкісного доступу до мережі Інтернет;
2) створення віртуальних приватних мереж (VPN);
3) передача голосу поверх IP;
4) проведення відеоконференцій;
5) захист інформації та зберігання даних;

## 1.3 Переваги і можливості корпоративної мережі

Впровадивши на своєму підприємстві корпоративну мережу, ви зможете організувати і чітко налагодити взаємодію між різними офісами, перетворивши їх в єдину систему. Розрізнені філії, розташовані на відстані офіси або навіть склади

і торгові точки, будучи об'єднаними загальною мережею, отримують ряд важливих переваг.

В першу чергу, це єдиний інформаційний простір . Єдина офісна мережа, що зв'язує всі майданчики, дає можливість зробити роботу компанії більш ефективною. Обмін інформацією, передача доручень і звітів, оформлення різної документації стають швидшими і зручнішими.

Друга важлива перевага - єдина система документообігу . Відправлення та прийом документів можуть здійснюватися по всіх каналах мережі, або тільки по обраним, в залежності від ступеня секретності.

Доступ до файлів, баз даних, архівів, підключення друкуючих пристроїв і т.д.. організовується дистанційно . Всі документи можна звести в чітко структуровану інформаційну базу, завдяки якій процес документообігу прискориться і спроститься.

Якщо коротко, в список переваг можна включити наступні пункти:

1) діяльність компанії стає прозорою для керуючих кадрів і керівництва;

2) роботу всіх структурних підрозділів організації можна контролювати якісно і оперативно;

3) внутрішній доступ до всіх баз даних, документації і звітності здійснюється в режимі реального часу;

4) локальна мережа істотно заощаджує витрати на міжміські, міжнародні дзвінки та кур'єрські послуги;

## 1.4 Складнощі при створенні корпоративної мережі

Основною трудністю (втім, цілком можливо розв'язати!) При влаштуванні мереж стає вибір оптимального способу, за допомогою якого буде створена єдина система. Це залежить від відстані між робочими майданчиками, необхідної швидкості передачі даних, провайдерських послуг і сервісів, які потрібні замовнику, а також ряду інших факторів.

Пристрій мережі може здійснюватися за допомогою як кабельних, так і бездротових систем. Як середовище для передачі даних виступає Інтернет, або система орендованих каналів (наприклад, з використанням VPN тунелів). Кожен із способів має свої недоліки і переваги.

Пристрій мережі із застосуванням бездротового обладнання дозволяє знизити витрати на експлуатацію мережі. Плата за абонентське обслуговування мережі, як правило, не стягується. Зв'язок між робочими майданчиками здійснюється за допомогою бездротового мосту або маршрутизатора.

До недоліками бездротового зв'язку можна віднести те, що для якісної передачі даних бажана "пряма видимість" майданчиків відносно один одного. На великих відстанях швидкість передачі даних може відчутно зменшаться.

Використовуючи в якості середовища для передачі даних Інтернет , компанія може знизити свої витрати на абонентську плату. Це економічний і доступний спосіб, тим більше що інтернет в даний час є практично скрізь. Однак така система не володіє високою надійністю, а швидкість передачі даних не буде стабільною і гарантовано високою.

Найбільш надійним способом вважається побудова мережі на основі орендованих каналів. Використовуючи мультисервісну мережу від того чи іншого провайдера, ви отримаєте гарантовано високу швидкість передачі даних, надійність з'єднання, а також зможете користуватися великим набором сервісів і послуг, які надає провайдер

Число користувачів і комп'ютерів в корпоративній мережі може вимірюватися тисячами, а число серверів - сотнями; відстані між мережами окремих територій можуть виявитися такими, що використання глобальних зв'язків стає необхідним .

Для з'єднання віддалених локальних мереж і окремих комп'ютерів в корпоративній мережі застосовуються різноманітні телекомунікаційні засоби, в тому числі канали первинних мереж, радіоканали, супутниковий зв'язок.

Неодмінним атрибутом такої складної і великомасштабної мережі є висока ступінь неоднорідності (гетерогенності) - не можна задовольнити потреби тисяч

користувачів за допомогою однотипних програмних і апаратних засобів. У корпоративної мережі обов'язково задіють різні типи комп'ютерів - від мейнфреймів до персональних комп'ютерів, кілька типів операційних систем і безліч різних додатків. Неоднорідні частини корпоративної мережі повинні працювати як єдине ціле, надаючи користувачам по можливості зручний і простий доступ до всіх необхідних ресурсів

## 1.5 Аналіз відомих методів і засобів

Високопродуктивні розподілені обчислення. Важливим класом розподілених систем є система, що використовується для високопродуктивних обчислювальних завдань. Грубо кажучи, можна розрізнити дві підгрупи.

У кластерних обчисленнях базове обладнання складається з колекції подібних робочих станцій або ПК, тісно пов'язаних за допомогою високошвидкісної локальної мережі. Крім того, на кожному вузлі працює одна і та ж операційна система.

Ситуація стає зовсім іншою у випадку мережевих обчислень . Ця підгрупа складається з розподілених систем, які часто будуються як об'єднання комп'ютерних систем, де кожна система може підпадати під інший адміністративний домен і може сильно відрізнятися, коли мова йде про обладнання, програмне забезпечення та розгорнуту мережеву технологію.

З точки зору сіткових обчислень, наступним логічним кроком є просто передати всю інфраструктуру, необхідну для обчислювальних додатків, на аутсорсинг . По суті, саме в цьому полягає хмарні обчислення : надання засобів для динамічного побудови інфраструктури та складання необхідного з доступних послуг.На відміну від мережевих обчислень, які тісно пов'язані з високопродуктивними обчисленнями, хмарні обчислення - це набагато більше, ніж просто надання великої кількості ресурсів.

Високопродуктивні обчислення більш-менш почалися із впровадження багатопроцесорних машин . У цьому випадку декілька процесорів організовані

таким чином, що всі вони мають доступ до однієї і тієї ж фізичної пам'яті, як показано на рис. 1.1

На відміну від цього, у мультикомп’ютерній системі декілька комп’ютерів під’єднані через мережу, і немає спільного використання основної пам’яті, як показано на рис. 5 b. Існують різні способи досягнення цього спільного доступу до основної пам’яті, але це є менш важливим у світлі нашої дискусії зараз. Більш важливим є те, що модель спільної пам'яті виявилася надзвичайно зручною для підвищення продуктивності програм, і її було відносно легко програмувати.

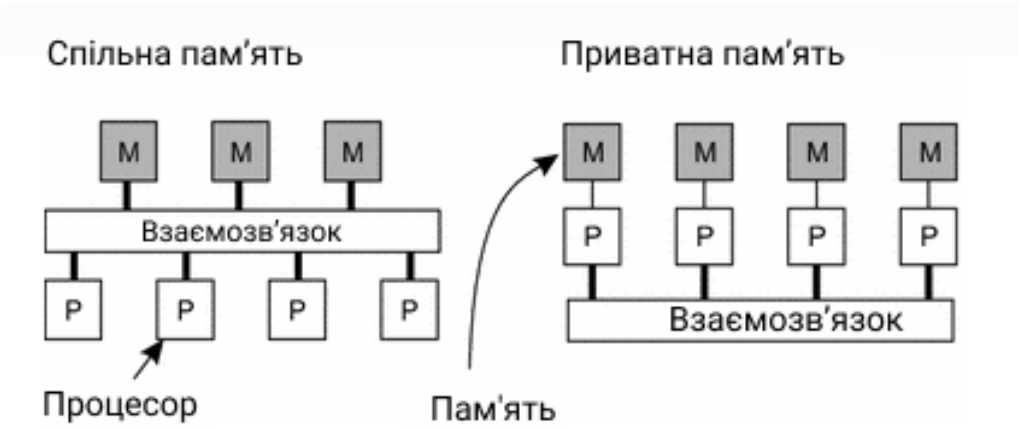



Рисунок 1.1 - Багатопроцесорна архітектура в порівнянні з багатокомп'ютерною архітектурою

Суть паралельних програм спільної пам'яті полягає в тому, що декілька потоків управління виконуються одночасно, тоді як усі потоки мають доступ до спільних даних. Доступ до цих даних контролюється через добре зрозумілі механізми синхронізації, такі як семафори. На жаль, модель нелегко масштабувати: на сьогодні розроблено машини, в яких лише кілька десятків процесорів мають ефективний доступ до спільної пам'яті. Певною мірою ми спостерігаємо однакові обмеження для багатоядерних процесорів, деякі з яких є багато процесорними, але деякі з них не є.

Щоб подолати обмеження систем спільної пам'яті, високопродуктивні обчислення перейшли на системи розподіленої пам'яті. Цей зсув також означав, що

багатьом програмам доводилося використовувати передачу повідомлень замість модифікації спільних даних як засобу зв'язку та синхронізації між потоками. На жаль, моделі передачі повідомлень виявились набагато складнішими та схильні до помилок порівняно з моделями програмування із спільною пам'яттю. З цієї причини було проведено значні дослідження в спробах побудови так званих розподілених спільних пам’яті мультикомп’ютерів або просто системи DSM.

По суті, система DSM дозволяє процесору звертатися до місця пам'яті на іншому комп'ютері так, ніби це локальна пам'ять. Цього можна досягти за допомогою існуючих методів, доступних для операційної системи, наприклад, шляхом зіставлення всіх сторінок основної пам'яті різних процесорів в єдиний віртуальний адресний простір. Кожного разу, коли процесор AA звертається до сторінки, розташованої на іншому процесорі BB, сталася помилка сторінки на AA дозволяючи операційній системі на AA щоб отримати вміст сторінки, на яку посилається BB таким же чином, як правило, він завантажує його локально з диска. У той же час процесор BB буде повідомлено, що наразі сторінка недоступна.

Від цієї елегантної ідеї імітації систем спільної пам’яті за допомогою мульти-комп’ютерів зрештою довелося відмовитись з тієї простої причини, що продуктивність ніколи не могла відповідати очікуванням програмістів, які скоріше вдавались би до набагато складніших, але краще (передбачувано) моделей програмування передачі повідомлень. .

Важливим побічним ефектом дослідження апаратно-програмних меж паралельної обробки є ретельне розуміння моделей узгодженості.

## 1.6 Кластерні обчислення

Кластерні обчислювальні системи стали популярними, коли співвідношення ціни продуктивності персональних комп'ютерів та робочих станцій покращилося. У певний момент стало фінансово та технічно привабливим побудувати суперкомп'ютер, використовуючи готову технологію, просто підключивши колекцію відносно простих комп'ютерів у високошвидкісній мережі. Практично у всіх випадках кластерні обчислення використовуються для паралельного програмування, в якому паралельно запускається одна (обчислювальна) програма на декількох машинах.

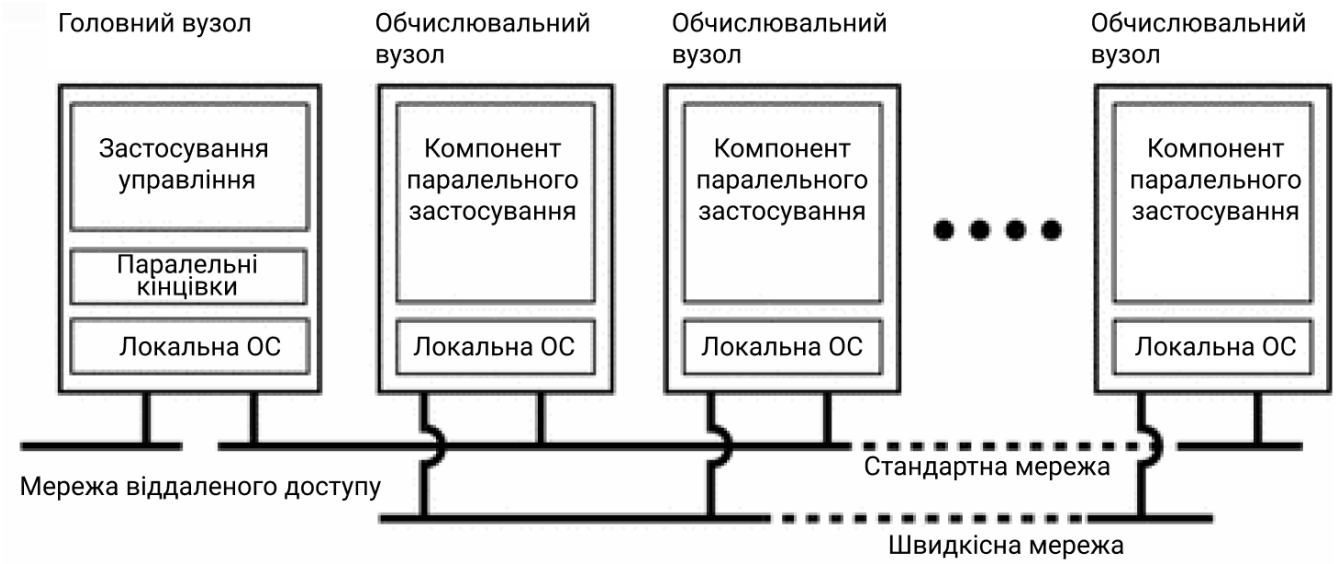



Рисунок 1.2 - Приклад кластерної обчислювальної системи

Аналіз відомих методів і засобів Одним із широко застосовуваних прикладів кластерного комп'ютера є кластери Beowulf на базі Linux, загальна конфігурація яких показана на рис. 1.2 Кожен кластер складається з колекції обчислювальних вузлів, які контролюються та до яких здійснюється доступ за допомогою одного головного вузла. Майстер, як правило, обробляє розподіл вузлів для певної паралельної програми, підтримує пакетну чергу поданих завдань та забезпечує інтерфейс для користувачів системи. Таким чином, майстер фактично запускає проміжне програмне забезпечення, необхідне для виконання програм та управління

кластером, тоді як обчислювальні вузли оснащені стандартною операційною системою, розширеною типовими функціями проміжного програмного забезпечення для зв'язку, зберігання, відмовостійкості тощо. Окрім головного вузла, обчислювальні вузли, таким чином, вважаються дуже ідентичними.

Ще більш симетричний підхід дотримується в системі MOSIX. MOSIX намагається надати односистемний образ кластера, що означає, що для процесу кластерний комп'ютер пропонує максимальну прозорість розподілу, видаючись єдиним комп'ютером. Як ми вже згадували, надати таке зображення за будь-яких обставин неможливо. У випадку з MOSIX високий ступінь прозорості забезпечується, дозволяючи процесам динамічно та превентивно мігрувати між вузлами, що складають кластер. Міграція процесів дозволяє користувачеві запускати програму на будь-якому вузлі (іменованому як домашній вузол), після чого він може прозоро переходити до інших вузлів, наприклад, для ефективного використання ресурсів. Подібні підходи при спробі створити односистемне зображення порівнюють Lottiaux et al.

Однак кілька сучасних кластерних комп'ютерів відійшли від цих матричних архітектур до більш гібридних рішень, в яких проміжне програмне забезпечення функціонально розподілено між різними вузлами. Перевага такого розділення очевидна: наявність обчислювальних вузлів із виділеними легкими операційними системами, швидше за все, забезпечує оптимальну продуктивність для обчислювальних програм. Подібним чином, функціональні можливості зберігання можуть, швидше за все, оптимально оброблятися іншими спеціально налаштованими вузлами, такими як файлові та сервери каталогів. Те саме стосується інших спеціалізованих послуг проміжного програмного забезпечення, включаючи управління роботою, послуги баз даних і, можливо, загальний доступ до Інтернету до зовнішніх служб.

## 1.7 Сіткові обчислення

Характерною особливістю традиційних кластерних обчислень є її однорідність. У більшості випадків комп’ютери в кластері здебільшого однакові, мають однакову операційну систему та всі підключені через одну мережу. Однак, як ми щойно обговорювали, спостерігається тенденція до створення більш гібридних архітектур, в яких вузли спеціально облаштовані для певних завдань. Це різноманіття є ще більш поширеним у мережевих обчислювальних системах: не робиться жодних припущень щодо подібності обладнання, операційних систем, мереж, адміністративних доменів, політики безпеки тощо.

Ключовим питанням у системі обчислювальних мереж є те, що ресурси різних організацій об’єднуються, щоб забезпечити співпрацю групи людей з різних установ, що фактично утворює федерацію систем. Така співпраця реалізується у формі віртуальної організації . Процеси, що належать одній і тій же віртуальній організації, мають права доступу до ресурсів, що надаються цій організації. Як правило, ресурси складаються з обчислювальних серверів (включаючи суперкомп'ютери, можливо реалізовані як кластерні комп'ютери), сховищ та баз даних. Крім того, також можуть бути передбачені спеціальні мережеві пристрої, такі як телескопи, датчики тощо.

З огляду на його природу, велика частина програмного забезпечення для реалізації мережевих обчислень розвивається навколо забезпечення доступу до ресурсів з різних адміністративних доменів та лише до тих користувачів та додатків, які належать до певної віртуальної організації. З цієї причини увага часто приділяється архітектурним питанням. Архітектура, спочатку запропонована Фостером. Показано на рис.  1.3. , що досі є основою для багатьох сіткових обчислювальних систем.


Рисунок 1.3 - Багатошарова архітектура для обчислювальних систем сітки

Зазвичай колектив, рівень зв’язку та рівень ресурсів становлять серцевину того, що можна назвати мережевим проміжним шаром. Ці рівні спільно забезпечують доступ до ресурсів та управління ними, які потенційно розподіляються між кількома сайтами.Важливим зауваженням з точки зору проміжного програмного забезпечення є те, що в обчислювальних мережах поняття сайту (або адміністративної одиниці) є загальним. Ця поширеність підкреслюється поступовим переходом до архітектури, орієнтованої на послуги, в якій сайти пропонують доступ до різних рівнів через колекцію веб-служб. Це на сьогоднішній день призвело до визначення альтернативної архітектури, відомої як архітектура Open Grid Services Architecture ( OGSA ). OGSI базується на оригінальних ідеях, сформульованих Фостером  проте, пройшовши процес стандартизації, він, як мінімум, робить його складним. Впровадження OGSA, як правило, відповідає стандартам веб-сервісу.

## 1.8 Хмарні обчислення

Поки дослідники розмірковували над тим, як організувати легко доступні обчислювальні мережі, організації, відповідальні за роботу центрів обробки даних, стикалися з проблемою відкриття своїх ресурсів для клієнтів.

Зрештою, це призвело до концепції обчислювальних програм, за допомогою якої замовник міг завантажувати завдання до центру обробки даних і отримувати плату за кожен ресурс

Службові обчислення лягли в основу того, що зараз називають хмарними обчисленнями .

Слідом за Вакеро, хмарні обчислення характеризуються зручним та доступним пулом віртуалізованих ресурсів.

Які та як використовуються ресурси можна динамічно налаштовувати, створюючи основу для масштабованості: якщо потрібно зробити більше роботи, клієнт може просто придбати більше ресурсів. Посилання на обчислювальні програми формується тим, що хмарні обчислення, як правило, базуються на моделі оплати за використання, в якій гарантії надаються за допомогою індивідуальних угод про рівень обслуговування (SLA).

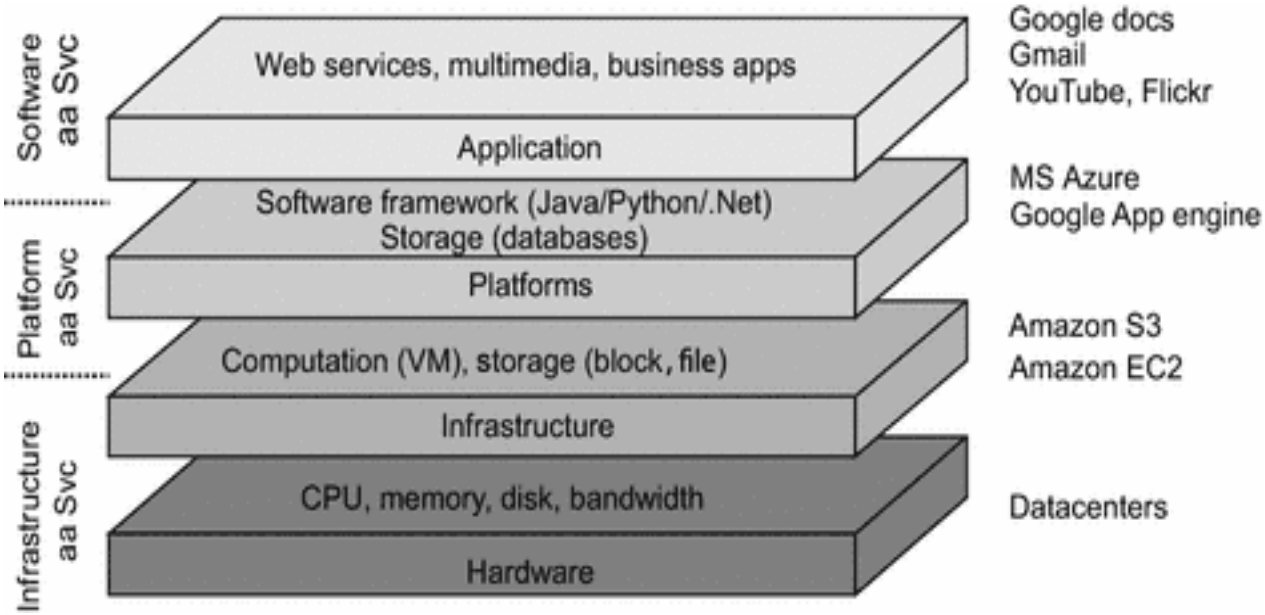



Рисунок 1.4 - Організація хмар

Постачальники хмарних обчислень пропонують ці рівні своїм клієнтам через різні інтерфейси (включаючи інструменти командного рядка, інтерфейси програмування та веб-інтерфейси), що веде до трьох різних типів послуг:

Інфраструктура як послуга ( мтС ), що охоплює апаратний та інфраструктурний рівень. Платформа як послуга ( РаМС ), що охоплює рівень платформи. Програмне забезпечення як послуга ( Sa MC ), в якому охоплюються їх додатки

На сьогоднішній день користуватися хмарами порівняно просто. Як наслідок, хмарні обчислення як засіб аутсорсингу локальних обчислювальних інфраструктур стали серйозним варіантом для багатьох підприємств. Однак існує ще низка серйозних перешкод, включаючи блокування постачальника, проблеми безпеки та конфіденційності, а також залежність від доступності послуг, якщо згадати деякі. Крім того, оскільки подробиці про те, як насправді виконуються конкретні хмарні обчислення, загалом приховані, і навіть, можливо, невідомі або непередбачувані, задоволення вимог щодо продуктивності може бути неможливим заздалегідь домовитись. Крім цього, Li et al. показали, що різні провайдери можуть легко показувати дуже різні профілі продуктивності. Хмарні обчислення вже не є

ажіотажем, і, безумовно, серйозною альтернативою підтримці величезної локальної інфраструктури, проте є ще багато можливостей для вдосконалення.

## 1.9 Організація розподіленої системи

Інший важливий клас розподілених систем є в організаціях, які стикаються з безліччю мережевих додатків, але для яких взаємодія виявилася болючим досвідом. Багато існуючих рішень проміжного програмного забезпечення є результатом роботи з інфраструктурою, в якій було простіше інтегрувати додатки в загально корпоративну інформаційну систему

Ми можемо виділити кілька рівнів, на яких може відбуватися інтеграція. У багатьох випадках мережева програма просто складається із сервера, що запускає цю програму (часто включаючи базу даних) і зробити її доступною для віддалених програм, які називаються клієнтами .

Такі клієнти надсилають на сервер запит на виконання певної операції, після чого відповідь надсилається назад. Інтеграція на найнижчому рівні дозволяє клієнтам повернути декілька запитів, можливо, для різних серверів, в один більший запит і виконати його як розподілену транзакцію . Ключова ідея полягає в тому, що всі або жоден із запитів не виконується.

Оскільки додатки стали більш досконалими і поступово були розділені на незалежні компоненти (зокрема, відрізняючи компоненти бази даних від обробних компонентів), стало ясно, що інтеграція також повинна відбуватися, дозволяючи програмам взаємодіяти безпосередньо між собою. Зараз це призвело до величезної галузі, яка концентрується на інтеграції корпоративних додатків ( EAI ).

## 1.10 Інтеграція корпоративних додатків

Як уже згадувалося, чим більше додатків відокремлювалося від баз даних, на яких вони були побудовані, тим очевидным стало те, що необхідні засоби для інтеграції програм незалежно від їх баз даних. Зокрема, компоненти програми

повинні мати можливість взаємодіяти безпосередньо між собою, а не лише за допомогою поведінки запиту / відповіді, що підтримується системами обробки транзакцій.

Ця потреба у взаємозастосунковому спілкуванні призвела до багатьох різних моделей спілкування. Основною ідеєю було те, що існуючі програми могли безпосередньо обмінюватися інформацією, як показано на рис. 1.5

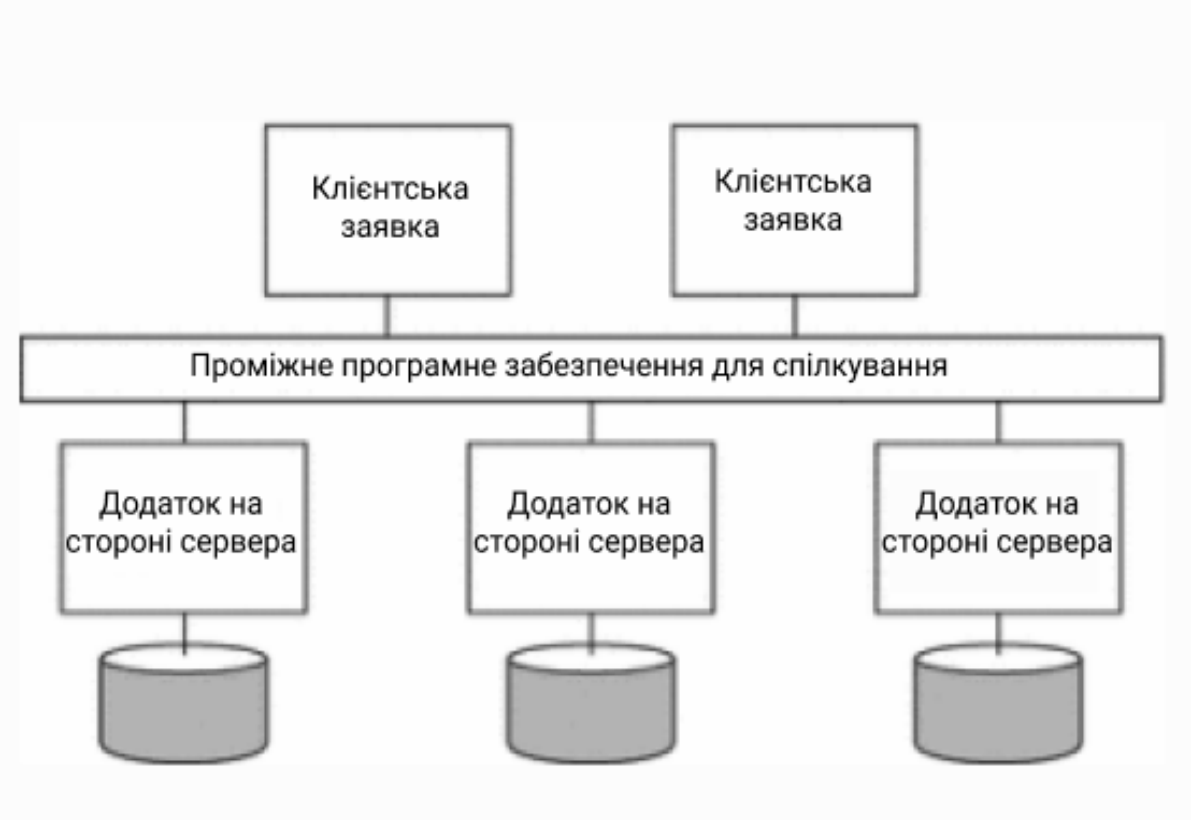



Рисунок 1.5 - Проміжне програмне забезпечення як посередник спілкування при інтеграції корпоративних програм

Існує кілька типів проміжного програмного забезпечення для зв'язку. За допомогою віддалених викликів процедур ( RPC ) компонент програми може ефективно відправляти запит іншому компоненту програми, виконуючи виклик локальної процедури, в результаті чого запит упаковується як повідомлення та надсилається абоненту. Аналогічно, результат буде відправлений назад і повернутий до програми як результат виклику процедури.

У міру того, як популярність об’єктної технології зростала, були розроблені методи, що дозволяють здійснювати виклики до віддалених об’єктів, що призводить до так званого віддаленого виклику методу ( RMI ). RMI по суті те ж саме, що і RPC, за винятком того, що він працює на об'єктах, а не на функціях.

RPC та RMI мають той недолік, що абоненту та абоненту потрібно бути готовим до роботи під час спілкування. Крім того, вони повинні точно знати, як посилатися один на одного. Це щільне з'єднання часто сприймається як серйозний недолік і призвело до того, що називається проміжним програмним забезпеченням , орієнтованим на повідомлення , або просто МОМ . У цьому випадку програми надсилають повідомлення логічним контактним точкам, які часто описуються за допомогою теми. Подібним чином програми можуть вказувати свою зацікавленість у певному типі повідомлень, після чого проміжне програмне забезпечення спілкування подбає про те, щоб ці повідомлення надходили до цих програм. Ці так звані системи публікації / передплати утворюють важливий і розширюваний клас розподілених систем.

Підтримка інтеграції корпоративних програм є важливою метою багатьох продуктів проміжного програмного забезпечення. Загалом існує чотири способи інтеграції програм.

## 1.11 Передача файлів

Суть інтеграції за допомогою передачі файлів полягає в тому, що програма створює файл, що містить спільні дані, які згодом читаються іншими програмами. Підхід технічно дуже простий, що робить його привабливим. Однак недоліком є те, що є багато речей, про які потрібно узгодити:

Формат і макет файлу Текст, двійковий файл, його структура тощо. У наш час XML став популярним, оскільки його файли в принципі самоописуються.

Управління файлами, де вони зберігаються, як їх називають, хто відповідає за видалення файлів?

Поширення оновлення Коли програма створює файл, можливо, існує кілька програм, яким потрібно прочитати цей файл, щоб забезпечити вигляд єдиної цілісної системи. Як наслідок, іноді потрібно застосовувати окремі програми, які повідомляють програми про оновлення файлів.

## 1.12 Спільна база даних

Багато проблем, пов’язаних з інтеграцією через файли, полегшуються при використанні спільної бази даних. Усі програми матимуть доступ до одних і тих самих даних, і часто за допомогою мови високого рівня, такої як SQL. Крім того, програми легко повідомляти про зміни, оскільки тригери часто є частиною сучасних баз даних. Однак є два основні недоліки. По-перше, все ще існує потреба у розробці загальної схеми даних, яка може бути далеко не тривіальною, якщо набір програм, які потрібно інтегрувати, не є повністю відомим заздалегідь. По-друге, коли багато читань та оновлень, спільна база даних може легко стати вузьким місцем для продуктивності.Інтеграція виклику віддаленої процедури через файли або базу даних неявно передбачає, що зміни, які здійснює одна програма, можуть легко спровокувати інші програми для вжиття заходів. Однак практика показує, що іноді невеликі зміни насправді повинні викликати багато додатків до дії. У таких випадках важливим є насправді не зміна даних, а виконання ряду дій. Серію дій найкраще фіксувати шляхом виконання процедури (що, в свою чергу, може призвести до різного роду змін у спільних даних)

Щоб запобігти тому, що кожна програма повинна знати всі внутрішні елементи цих дій (як реалізовано іншою програмою), слід використовувати стандартні методи інкапсуляції, розгорнуті за допомогою традиційних викликів процедур або викликів об'єктів. У таких ситуаціях програма може найкраще запропонувати процедуру для інших програм у формі віддаленого виклику процедури або RPC.

По суті, RPC дозволяє додаток  AA використовувати інформацію, доступну лише для програми BB, не даючи AA прямий доступ до цієї інформації.

## 1.13 Обмін повідомленнями

Основним недоліком RPC є те, що для того, щоб дзвінок був успішним, потрібно одночасно активувати абонента, що телефонує, і того, хто викликає. Однак у багатьох сценаріях цю одночасну діяльність часто важко або неможливо гарантувати. У таких випадках пропонується система обміну повідомленнями, що містить запити від програми AA виконати дію при застосуванні BB, - це те, що потрібно. Система обміну повідомленнями гарантує, що в кінцевому підсумку запит буде доставлений, і, якщо потрібно, відповідь повернеться також. Очевидно, що обмін повідомленнями не є панацеєю для інтеграції додатків: він також створює проблеми, що стосуються форматування та розмітки даних, він вимагає, щоб програма знала, куди надсилати повідомлення, повинні існувати сценарії роботи з втраченими повідомленнями тощо.

## 1.14 Поширені системи

Обговорювані до цього часу розподілені системи в значній мірі характеризуються своєю стабільністю: вузли фіксовані і мають більш-менш постійне та якісне з'єднання з мережею. Певною мірою ця стабільність реалізується завдяки різним методам досягнення прозорості розподілу. Наприклад, є багато способів, як ми можемо створити ілюзію, що лише зрідка компоненти можуть вийти з ладу. Подібним чином, існують всілякі способи приховати фактичне мережеве розташування вузла, ефективно дозволяючи користувачам та програмам вважати, що вузли залишаються на місці.

Однак після впровадження мобільних та вбудованих обчислювальних пристроїв справи змінилися, що призвело до того, що зазвичай називають загально проникними системами . Як випливає з назви, всепроникне системи призначені природним чином вливатися в наше середовище. Вони, природно, також розподілені системи.

Багато пристроїв у всепроникаючих системах характеризуються тим, що вони малі, мають акумуляторні батареї, є мобільними та мають лише бездротове підключення, хоча не всі ці характеристики стосуються всіх пристроїв. Це не обов'язково обмежувальні характеристики, як ілюструють смартфони . Тим не менш, особливо той факт, що нам часто доводиться мати справу з тонкощами бездротового та мобільного зв'язку, вимагатиме спеціальних рішень, щоб зробити повсюдну систему якомога прозорішою або ненав'язливо.

## 1.15 Системи виявлення вторгнень

Процес виявлення та реагування на шкідливу діяльність, спрямовану на обчислювальні та мережеві ресурси. Це пристрій, як правило, інший комп’ютер, який контролює діяльність з ідентифікації шкідливих або підозрілих подій. IDS отримує вихідні вхідні дані від датчиків, аналізує ці вхідні дані, а потім робить певні дії. Оскільки витрати на обробку інформації та доступ до Інтернету падають, все більше організацій стають вразливими до широкого спектру кіберзагроз. Згідно з недавнім опитуванням CERT, останнім часом рівень кібератак щороку подвоюється.Тому стає все більш важливим зробити наші інформаційні системи, особливо ті, що використовуються для таких важливих функцій, як митарі та комерційні цілі, стійкими до таких атак і терпимими. Системи виявлення вторгнень (IDS) є невід’ємною частиною будь-якого пакету безпеки сучасної мережевої інформаційної системи. IDS виявляє вторгнення, відстежуючи мережу або систему та аналізуючи потік аудиту, зібраний із мережі або системи, щоб знайти підказки зловмисної поведінки відредагований том проливає нове світло на системи оповіщення про захист від вторгнень комп’ютерів та мереж. Він також охоплює інтеграцію попереджень про вторгнення в рамки політики безпеки для реагування на вторгнення, відповідні тематичні дослідження та багато іншого. Цей том поданий у легкодоступним стилі, включаючи сувору обробку питань, рішень та технологій, пов’язаних із галуззю Системи виявлення вторгнень розроблені для професійної аудиторії, що складається з дослідників та практиків в галузі

комп’ютерних мереж та галузі захисту інформації. Він також підходить як довідковий або додатковий підручник для студентів просунутого рівня з інформатики.Говорячи про IDS, що базується на хості, слід зупинитися на двох основних модулях: модулі збору даних та модулі аналізу. Склавши разом, вони визначають особливості цілого IDS. У першому розділі ми обговорюємо модуль аналізу. Гарний огляд попередніх робіт, присвячених проектуванню аналітичного модуля. Набір моделей опису вторгнень використовується модулем оцінки і визначає ймовірність небажаних дій, які система здатна виявити. Безліч атак, не описаних моделями, визначає повноту виявлення IDS, пов'язану з можливістю виявлення порушень безпеки. Запропонований підхід видається нам перспективним для розробки практичних IDS на основі хоста. У другому розділі обговорюються принципи проектування модуля збору даних на основі хоста. Базові роботи, пов’язані зі створенням модуля збору даних IDS на основі хоста, можна знайти в для Windows і в для Linux. Ми систематизуємо запропоновані методи проектування та аналізуємо їх характеристики.

## 1.16 Висновки

Незалежно від розміру вашого бізнесу, якщо потрібно підключити свої комп’ютери, принтери та будь-яке інше обладнання до локальної мережі. Налаштування корпоративної мережі дозволить ділитися інформацією у бізнесі. Крім того, можна ділитися тим обладнанням, яке у вас є. Наприклад, один принтер може обслуговувати кількох співробітників. Мережа не обмежується однією будівлею, яку займає ваш бізнес. Якщо в районі декілька будівель, можна створити мережу містечка для зв’язку працівників та ресурсів у кожній будівлі. Якщо ваш бізнес охоплює більше одного міста, декількох штатів або навіть якщо він розповсюджений по всьому світу, ваша корпоративна мережа може бути підключена за допомогою широкосмугової мережі або глобальної мережі.

# 2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ КОМП‘ЮТЕРНОЇ МЕРЕЖІ

Для реалізації конкретної корпоративної мережі буде використовуватись прогресивний підхід до побудови мережі відповідно до ієрархічної моделі комп'ютерної мережі запропонованої фірмою Cisco System, яка передбачає наявність трьох основних рівнів ККС рис.2.1

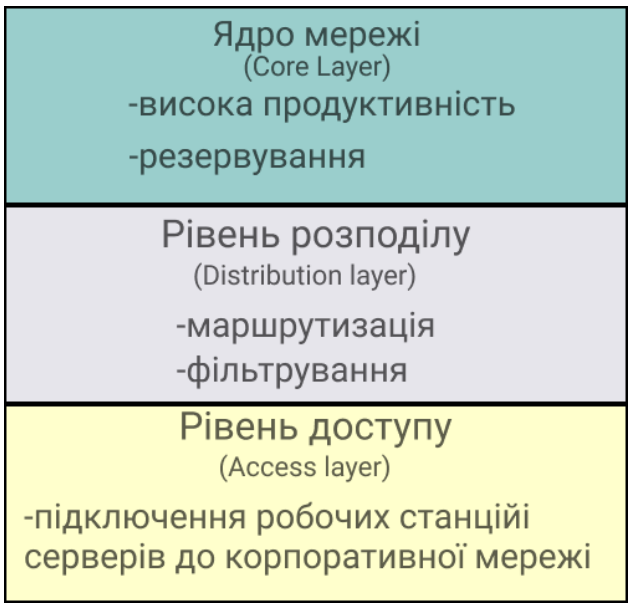



Рисунок 2.1 - Ієрархічна модель ККС

Перший рівень Core layer він створюється комутаторами, працюючими на другому і третьому рівнях відповідно до моделі взаємодії відкритих систем (OSI). Головним і єдиним призначенням рівня ядра є швидка комутація мережевого трафіку.

Трафік передається на даному рівні спільно для кількох користувачів. Однак на рівні розподілу обробляються призначені для користувача дані, що може призвести до додаткових запитах в базовий рівень. Якщо відбувається помилка на рівні ядра, то вона впливає на всіх користувачів. Отже, дуже важливо забезпечити високу надійність на базовому рівні. На цьому рівні обробляються великі обсяги трафіку, тому не менш важливо враховувати швидкість і затримки.

Другий рівень це рівень розподілу (Distribution layer) даний рівень утворюється комутаторами, які працюють на третьому і четвертому рівнях відповідно до моделі взаємодії відкритих систем (OSI). На цьому рівні здійснюється маршрутизація трафіку відповідно до заданих політиками і пріоритетами.

Основні функції рівня розподілу складаються в маршрутизації, фільтрації і доступі до регіональних мереж, а також (якщо необхідно) в визначенні правил доступу пакетів до базового рівня. Рівень розподілу зобов'язаний встановлювати найбільш швидкий спосіб обробки запитів до службам (наприклад, метод файлового звернення до сервера). Після визначення на рівні розподілу найкращого шляху доступу, запит може бути переданий на базовий рівень, де реалізований швидкісний транспорт запиту до потрібної службі. На рівні розподілу встановлюється політика мережі, а також забезпечуються можливості гнучкого опису мережевих операцій.

На рівні розподілу виконується такі функції:

1) реалізація інструментів, подібних списками доступу, фільтрації пакетів або механізму запитів;

2) реалізація системи безпеки і мережевих політик, включаючи трансляцію адрес і установку брандмауерів;

3) перерозподіл між протоколами маршрутизації, включаючи використання статичних шляхів;

4) маршрутизація між мережами VLAN і інші функції підтримки робочих груп;

5) визначення доменів широкомовних і багатоадресних розсилок.

Третій рівень (Access layer) рівень доступу даний рівень утворюється комутаторами, які працюють на другому рівні відповідно до моделі взаємодії відкритих систем (OSI). На цьому рівні реалізовано управління користувачами і робочими групами при зверненні до ресурсів об'єднаної мережі.Іноді рівень доступу називають рівнем настільних систем.

Найбільша частина необхідних користувачам мережевих ресурсів повинна бути доступна локально. Якщо на рівні розподілу виконується перенаправлення трафіку до віддалених службам. Для рівня доступу характерні наступні функції:

1) постійний контроль (з рівня розподілу) за доступом і політиками;
2) формування незалежних доменів конфліктів (сегментація);
3) з'єднання робочих груп з рівнем розподілу;
4) зазвичай на рівні доступу застосовуються технології DDR або комутація Ethernet;

Кожен рівень відповідає за реалізацію певних функцій. Однак ці рівні є логічними і не обов'язково погоджені з фізичними пристроями. Наприклад, в прийнятій ієрархічної моделі мережі OSI теж використовуються логічні рівні ієрархії, які описують всі функції мережі. При цьому певний протокол передачі даних не обов'язково відповідає певній функції і, отже, протокол може відображатися на декількох рівнях моделі OSI. Аналогічно, при побудові фізичної реалізації мережі - кілька пристроїв можуть потрапити на один рівень, або один пристрій буде виконувати функції декількох рівнів. Інакше кажучи, виділені нами рівні мережі є логічними, а не фізичними поняттями.

## 2.1 Вибір і аналіз комутаційного обладнання

Оскільки проектування мережі проводиться засобами імітаційного моделювання обчислювальних мереж Packet Tracer 6.0, розробленого компанією Cisco, для формування телекомунікаційних систем, організації бездротової мережі, IP телефонії і т. п. вибирається обладнання компанії Cisco Systems.

Вибір обладнання проводиться на основі певних вимог до того чи іншого обладнання. Згідно ієрархічної моделі побудови мереж, використовуваної в даній роботі, необхідно було вибрати обладнання для трьох рівнів мережі:

1) Рівня ядра.
2) Рівня розподілу.
3) Рівня доступу (Access layer).

Так само необхідно було вибрати маршрутизатор, який відповідає за підключення до інтернету, а також модулі SFP для підключення оптичних ліній зв'язку та бездротові точки доступу.

Рівень ядра і розподілу

Виходячи з фінансових і практичних міркувань, було прийнято рішення в якості рівня ядра і рівня розподілу використовувати один багаторівневий комутатор Catalyst 3560-G Cisco Catalyst WS-C3560G-24TS-S (рисунок 2.2).

Рисунок 2.2 - Багаторівневий комутатор Catalyst 3560-G Cisco Catalyst WS-C3560G-24TS-S

Серія Catalyst 3560 з фіксованою конфігурацією - на 24 порти Ethernet 10/100 / 1000T і 4 порта SFP. Версія ПО IPB Image конфігурацією (В конфігураціях FMC Ethernet и Gigabit Ethernet передбачено з'єднання пристроїв по кручений парі (PoE), сумісне зі специфікацією IEEE 802.3af і до-нестандартним варіантом Cisco). Основні особливості Cisco Catalyst 3560: потужна система управління якістю обслуговування (QoS); обмеження швидкості передачі даних; списки контролю доступу (ACL); управління мультивещание; високопродуктивна IP-маршрутизація. Технічні характеристики представлені в рисунку 2.3

| | |
|---|---|
| **Порти** | |
| Кількість портів | 24 |
| Фіксовані порти | 24 Ethernet 10/100/1000 ports |
| Аплінки | 4 SFP 1000 Mbps |
| **Характеристики** | |
| Рівень (L2, L3) | L3 |
| Пропускна здатність | 32 Gbps |
| Продуктивність (pps) | 38.7 Mpps |
| **Живлення** | |
| PoE | - |
| Блок живлення | AC internal power supply |

Рисунок 2.3 - Технічні характеристики

Рівень доступу

Основне завдання влаштування рівня доступу - надання доступу робочих станцій і серверів до наступного рівня (розподілу) ієрархії. У більшості випадків рівень доступу представлений в мережі комутаторами другого рівня. Вибір комутатора рівня доступу, ґрунтувався на таких критеріях, як оптимальна ціна і технічні характеристики.

Для корпоративної мережі були обрані комутатори другого рівня (L2)Catalyst WS-C2960G-24TC-L (рисунок 2.4) та Catalyst S-C2960G-48TC-LS,компа

Рисунок 2.4 - Коммутатор другого рівня Catalyst WS-C2960G-24TC-L

Інтелектуальні Ethernet-комутатори Cisco Catalyst 2960 – сімейство комутаторів другого рівня з фіксованою конфігурацією, яке дозволяє підключати робочі станції до мереж FMC Ethernet и Gigabit Ethernet на швидкості середовища передачі, задовольняючи зростаючі потреби в пропускної здатності на периферії мережі. Для агрегації застосовуються комбіновані гігабітні uplink-порти, які можуть об'єднуватися в єдиний канал за технологією Gigabit Ethernet Channel.

Для спрощення завдання конфігурації в комутаторах серії Catalyst 2960 передбачена функція Smartports, що дозволяє виконати основні налаштування порту комутаторів, грунтуючись на його призначені. Cisco Catalyst 2960 забезпечує потребу в передачі даних зі швидкістю 100 Мбіт / сек і 1 Гбіт / сек, дозволяють використовувати LAN сервіси, наприклад, для мереж передачі даних, побудованих у філіях корпорацій. Сімейство Catalyst 2960 дозволяє забезпечити високу безпеку даних за рахунок вбудованого NAC, підтримки QoS і високого рівня стійкості системи.

Маршрутизатор

У ККМ маршрутизатор виступає в ролі «воріт» у всесвітню мережу Інтернет. Вибір маршрутизатора грунтувався на його параметрах продуктивності і пропускної здатності каналів.

Для мережі підприємства був обраний маршрутизатор Cisco 2911: 3 порти 10/100/1000 mbiT Ethernet, 4 слота EHWIC, 2 слота PVDM3, 1 слот SM, 1слот ISM, блок живлення AC (Рис 2.5)

Рисунок 2.5 - Маршрутизатор Cisco 2911

Cisco 2911- серія мережевих маршрутизаторів з інтеграцією сервісів, розроблена спеціально для підприємств малого та середнього бізнесу на підставі 25-річного досвіду Cisco в області інновацій і створення передових рішень. Архітектура нових платформ забезпечує підтримку наступного етапу розвитку філій організацій, переносячи мультимедійні засоби спільної роботи і засоби віртуалізації на рівень філії і дозволяючи істотно скоротити операційні витрати.Платформи на базі маршрутизаторів Cisco ISR другого покоління дозволяють вирішувати не тільки сьогоднішні завдання, але і ті завдання, які виникнуть в майбутньому, оскільки в них використовуються багатоядерні процесори, підтримуються високопродуктивні сигнальні процесори (DSP) для розширення перспективних можливостей передачі відео, використовуються потужні сервісні модулі з підвищеною доступністю, засоби комутації Gigabit Ethernet з підтримкою розширеної специфікації POE, а також нові можливості управління і моніторингу споживання енергії. Крім того, новий універсальний образ операційної системи Cisco IOS® і модуль Services Ready Engine дозволяють розділити розгортання обладнання та програмного забезпечення, тим самим забезпечуючи надійну технологічну основу, здатну швидко адаптуватися до постійно мінливих вимог до мережі.

| | |
|---|---|
| Протоколи | IPv4, IPv6, статическая маршрутизация, OSPF, EIGRP, BGP, BGP Router Reflector, IS-IS, IGMPv3, PIM SM, PIM SSM, DVMRP, IPSec, GRE, BVD, механизмы групповой адресации IPv4-IPv6, MPLS, L2TPv3, 802.1ag, 802.3ah, L2 и L3 VPN |
| Інкапсуляція | Ethernet, 802.1q VLAN, соединение "точка-точка" (PPP), MLPPP, Frame Relay, MLFR (FR.15 и FR.16), HDLC, последовательные интерфейсы (RS-232, RS-449, X.21, V.35, и EIA-530), PPPoE и ATM |
| Управління трафіком | QoS, CBWFQ, WRED, средства иерархического обеспечения качества обслуживания, PBR, PfR и NBAR |

Рисунок 2.5 - Протоколи

Оптичні SFP модулі

SFP (англ. Small Form-factor Pluggable) невеликий приймач, який підключається до порту SFP мережевого комутатора і підключається до оптоволоконних кабелів Fibre Channel та Gigabit Ethernet (GbE) на іншому кінці. Замінюючи приймач GBIC, модулі SFP також називаються "міні-GBIC" через їх менший розмір. Вибравши відповідний модуль SFP, один і той же електричний порт на комутаторі може підключатися до волокон різної довжини та типу хвилі (багатомодовий або одномодовий). Якщо волокно оновлено, модуль SFP замінюється.

SFP перетворює послідовні електричні сигнали в послідовні оптичні сигнали і навпаки. Модулі SFP можна гаряче замінювати і містять ідентифікатор та інформацію про систему для комутатора

Рисунок 2.6 - Оптический SFP Cisco GLC-LH-SM

Cisco SFP (Small Form-factor Pluggable) призначений для установки в слот маршрутизатора або комутатора і забезпечують підключення його до мережі за допомогою потрібного інтерфейсу. Конвертори SFP підтримують режим гарячої заміни (hot-swap). Випускаються різні модулі, що дозволяють підключити необхідне обладнання до різних середовищ передачі: багатомодове оптоволокно, одномодовое оптоволокно, скручена пара.

| | |
|---|---|
| Вихідна потужність передавача | -9.5 ~ -3 dBm< |
| Чутливість прийомника | -22 dBm |
| Довжина хвилі передавача | 1270~1340 nm, (1310 nm) |
| Довжина хвилі прийомника | 1100~1600 nm, (1310 nm) |
| Швидкість передавання даних | 100Mbps~ 1,25Gbps |
| Максимальна довжина двоволкнового одномодового оптичного кабелю 9/125 мкм | 10 km |
| Робоча температура | 0~50 °C |
| Напруга | 3.3 V |
| Розєми для оптичного кабелю | двойной LC |

Рисунок 2.6 - Характеристики

Бездротова точка доступу

Для зв'язку з віддаленим ділянкою мережі буде застосовуватися бездротова технологія. Оскільки віддалени ділянка буде перебувати на відстані 500 метрів необхідно застосувати дві спрямовані антени для бездротових точок доступу. Головними критеріями при виборі бездротової точки доступу, були її дальність покриття, максимальна швидкість і наявність стандартів a / g / n. Вибір був зроблений на користь точки доступу Cisco AIR CAP2602I-A-K9 (рисунок 2.7).

Рисунок 2.7 - Бездротова точка доступу Cisco AIR CAP2602I-A-K9

Точки доступу серії Cisco Aironet 2600 добре підходять для корпоративних мереж будь-якого розміру, підтримують швидкість з'єднання до 450 Мбит / с працюють за стандартом 802.11n, підтримують MIMO за схемою 3x4, три просторових потоку, а також технології Cisco CleanAir ™, ClientLink 2.0 ™ і

VideoStream, що дозволяє забезпечувати високошвидкісну бездротову зв'язок без перешкод.

| | |
|---|---|
| **Загальна інформація** | |
| Виробник | Cisco |
| **загальні характеристики** | |
| Тип | Wi-Fi точка доступу |
| Стандарт бездротового зв'язку | 802.11 n, частота 2.4 / 5 ГГц |
| підтримка MIMO | є |
| Макс. швидкість бездротового з'єднання | 450 Мбіт / с |
| **Прийом передача** | |
| Захист інформації | WPA, WPA2, 802.1x |
| потужність передавача | 23 dBM |
| **Опції точки доступу / моста** | |
| швидкість портів | 10/100/1000 Мбіт / сек |
| **Моніторинг та конфігурування** | |
| консольний порт | є |
| Web-інтерфейс | є |
| **пам'ять** | |
| Об'єм оперативної пам'яті | 256 Мб |
| Обсяг флеш-пам'яті | 32 Мб |
| **додатково** | |
| Флеш-пам'ять | є |
| Харчування через Ethernet-кабель (PoE) | є |
| **Фізичні параметри (нетто)** | |
| Розміри (ШхВхГ) | 221x54x221 мм |
| Вага нетто) | 1040 г |

Рисунок 2.8 - Технічні характеристики

## 2.2 Вибір серверної інфраструктури мережі

Було вирішено, що в корпоративних комп'ютерних мережах використовується мережева архітектура клієнт-сервер (англ. Client-server)

Ця система дозволяє мережі уніфікувати функції та програми одного або декількох зарезервованих файлових серверів. Файлові сервери служать невід'ємною частиною системи та забезпечують безпеку та доступ до ресурсів.

У мережі клієнт / сервер користувачі або клієнти мають доступ до ресурсів, доступних на файловых серверах. Мережеві операційні системи дозволяють більш ніж кільком користувачам одночасно спільно використовувати однакові ресурси незалежно від фізичного місцезнаходження. Novell Netware, Windows 2000 Server та інші - найкращі приклади для клієнтських і серверних операційних систем. У мережі користувач може виконувати свої дії, входячи в операційні системи, подібні до локальної мережі. Спільний доступ до файлів, документів, апаратних ресурсів та виконання віддалених машин також можна завершити через NOS оскільки дослідники використовують переваги комп’ютерних та пов’язаних з ними технологій, обсяг даних різко зріс. Останні розробки в системах управління базами даних (СУБД), таких як технологія баз даних клієнт-сервер, забезпечують необхідні інструменти для злиття та управління даними з різних областей дослідження, а також з різних місцезнаходжень. Ця нова технологія дозволяє розподіляти операції обробки та зберігання даних на різноманітних комп’ютерних платформах, тим самим максимізуючи ефективність та гнучкість. Описана СУБД, яка використовує технологію клієнт-сервер. Для забезпечення повноцінної роботи підприємства та її бізнес процесів, необхідна установка наступних серверів:

Сервер системи ERP

ERP - це важливе програмне забезпечення для бізнесу, яке збирає інформацію з різних підрозділів у загальній базі даних, що дозволяє керівникам контролювати пульс компанії, використовуючи єдине бачення реальності.

Системи корпоративного планування ресурсів об'єднують такі важливі ділові функції, як фінанси, виробництво, управління запасами та замовленнями, комунікація з клієнтами, продажі та маркетинг, управління проектами та людські ресурси. Однією з основних особливостей є детальна аналітика та звітність щодо кожного відділу. ERP може генерувати значну економію часу та фінансів, забезпечуючи видимість в масштабах всієї організації, яка виявляє неефективні

процеси та відкриває можливості для зростання.Існує кілька моделей розгортання програмного забезпечення ERP, включаючи локальні, хмарні та гібридні. Хоча хмарна ERP стала надзвичайно популярною в останні роки який підхід найкраще залежить від потреб компанії.

В якості такої системи обрана Microsoft Dynamics NAV інтегрована з хмарним сервісом Office 365. До переваг даної системи можна віднести:

1) наявність функціональних інтегрованих між собою модулів, необхідних для повного управління діяльністю компанії (фінанси, логістика, виробництво, кадри і т.д.);

2) повну відповідність вимогам російського законодавства, що дозволяє використовувати Microsoft Dynamics NAV для ведення російського бухгалтерського, податкового обліку, розрахунку зарплати і підготовки всієї оперативної документації і регулярної звітності;

3) потужний фінансовий функціонал, за допомогою якого можна, в тому числі, розраховувати собівартість продукції і проводити всебічний аналіз діяльності організації;

4) підтримка різних варіантів доступу до системи, в тому числі віддалена робота через Інтернет або мобільні пристрої. така гнучкість особливо важлива для географічно розподілених компаній або підприємств, які використовують хмарну інфраструктуру;

5) зручність настройки і використання без зупинки роботи підприємства;

6) модифікації в систему можна вносити за допомогою вбудованого мови програмування C / AL.

Контролер домену

Установка контролера домену це важлива частина для комп'ютерної мережі, по суті, контролює її роботу. Його основне завдання  запуск важливої служби Active Directory.

Він працює з центром поширення ключів - Kerberos. Також передбачає роботу на Unix-сумісних системах. В них в якості контролера виступає комплект програмного забезпечення Samba. Контролер домену служить для створення

локальної мережі, в якій могли б авторизуватися користувачі під своїм ім'ям і зі своїми обліковими даними. Вони повинні це робити на всіх комп'ютерах. Також установка контролера домену забезпечує визначення права доступу в мережі і управління її безпекою. Таким чином, з його допомогою можна централізовано адмініструвати всі мережеві ресурси, включаючи користувачів, файли, периферійні пристрої, доступ до служб, мережевих ресурсів, веб узлам, баз даних.

Веб-сервер. Даний сервер необхідний для створення інформаційного порталу підприємства так званого інтранету. Інтранет - мережа приватних підприємств, призначена для підтримки працівників організації у спілкуванні, співпраці та виконанні своїх ролей. Він служить для широкого кола цілей і використання, але в основі своєї діяльності є інтранет, щоб допомогти працівникам . .

Сервер баз даних. Сервер баз даних, буде виконувати обслуговування та управління базою даних (СКБД) і відповідати за цілісність і збереження даних, а також забезпечувати операції введення-виведення при доступі клієнта до інформації.

Файл-сервер. Виділений сервер, призначений для виконання файлових операцій введення-виведення і зберігає файли будь-якого типу, що володіє великим об'ємом дискового простору, реалізований у формі RAID-масиву для забезпечення безперебійної роботи і підвищеної швидкості запису і читання даних.

Поштовий сервер. Для організації стабільної та надійної роботи корпоративної електронної пошти не обійтися без корпоративного почтового сервера, який дозволяє підтримувати один або декілька доменів організації, налаштувати надійний захист від спаму і вірусних атак, здійснювати контроль за корпоративною поштою і так далі. Зазвичай поштовий сервер працює «за кулісами», а користувачі мають справу з іншою програмою - клієнтом електронної пошти англ. mail user agent, MUA).

Сервер IP-телефонії. IP-телефонія - телефонний зв'язок по протоколу IP. Під IP-телефонією мається на увазі набір комунікаційних протоколів, технологій і методів, що забезпечують традиційні для телефонії набір номера, дзвінок і двостороннє голосове спілкування.

Рисунок 2.9 - Сервер HP Proliant BL660c Gen9

HP Proliant BL660c Gen9 (G9) - це блейд-сервер половинної висоти з підтримкою 4 процесорів і великого обсягу оперативної пам'яті. Сервер забезпечує чудову масштабованість і високу продуктивність блейд-технології, яка досягається застосуванням 18-ти ядерних процесорів. Блейд-сервер HP ProLiant BL660c Gen9 підтримує кілька контролерів для реалізації багаторівневої системи зберігання (HP Smart Array P246br і HP Dynamic Smart Array B140i), що підвищує гнучкість розгортання систем.

## 2.3. Вибір засобів моніторингу мережі.

Процес моніторингу наявності, тривалості роботи, роботи та продуктивності складних мереж. Це передбачає відстеження та аналіз мережевих компонентів, таких як маршрутизатори, комутатори та брандмауери, а також зв’язок між ними. Це також передбачає обстеження різних рівнів даних, кінцевих точок мережі та посилань.

Перевірка працездатності та продуктивності мережевих інтерфейсів на предмет їх несправностей допомагає діагностувати, оптимізувати та керувати різними мережевими ресурсами як на місці, так і віддалено. Маючи дані у вигляді

таблиць, діаграм, графіків, інформаційних панелей та звітів, моніторинг мережі допомагає IT-адміністраторам скоротити середній час на ремонт (MTTR) та вирішити проблеми з продуктивністю мережі в режимі реального часу. Наприклад, щоб визначити стан веб-сервера, програма моніторингу може час від часу надсилати запит на отримання сторінки. Для поштових серверів можна відправити тестове SMTP-повідомлення і отримати відповідь по протоколам POP3 або IMAP. Невдалі запити (з'єднання не може бути встановлено, воно завершується по тайм-ауту або, коли повідомлення не було доставлено) зазвичай викликають одну або кілька реакцій з боку системи моніторингу: відсилання сигналу тривоги системного адміністратора; автоматичне включення системи захисту від збоїв, яка тимчасово виведе проблемний сервер з експлуатації, постарається вирішити проблему за допомогою спеціальних скриптів і допоміжних програм. Існує безліч засобів моніторингу мережі: Cacti, Nagios, netSNMP і т.д. Для моніторингу була обрана система моніторингу The Dude.

The Dude - безкоштовна програма, призначена для сканування мереж, моніторингу роботи підключених до них пристроїв і попередження адміністратора в разі виникнення будь-яких проблем. Серед існуючих систем моніторингу The Dude позиціонується як кращий по співвідношенню ціна / якість продукт. Функціонал Dude дозволяє контролювати окремі сервери, а також мережі та мережеві сервіси будь-якого ступеня складності.

## 2.4 Розробка плану IP-адресації

IP-адреси (IP address) представляють собою основний тип адрес, на підставі яких мережевий рівень передає повідомлення, звані IP пакетами. Ці адреси складаються з 4 байт, записаних в десятковому вигляді та розділених точками, наприклад, 117.52.9.44. Номер вузла в протоколі IP призначається незалежно від локальної адреси вузла. Маршрутизатор по визначенню входить відразу в кілька мереж. Тому кожен порт маршрутизатора має власний IP-адреса. Кінцевий вузол також може входити в кілька IP-мереж. У цьому випадку комп'ютер повинен мати

кілька IP-адрес, по числу мережевих адаптерів. Таким чином, IP-адреса характеризує не окремий комп'ютер або маршрутизатор, а одне мережеве з'єднання. План IP-адресації є фундаментом для реалізації всього проекту мережі. Грамотно складений IP план дозволяє знизити навантаження на обладнання в разі великих територіально розподілених інсталяцій і спрощує розуміння інфраструктури обслуговуючим персоналом, що в свою чергу знижує ризики відмов елементів мережі через людський фактор. Для корпоративної була обрана приватна IP-адресація класу С з маскою підмереж 255.255.255.0, максимальна кількість комп'ютерів в одній підмережі 256.

## 2.5 Створення схеми комп'ютерної мережі

Весь процес розробки корпоративної мережі грунтувався на існуючою схемою інформаційних потоків, основою якої є сервер системи ERP. В ході розробки була підібрана відповідна топологія мережі, підібрано комутаційне і серверне обладнання, сервер IP-телефонії. В результаті опрацювання питання по підбору обладнання розробляється мережі підприємства була розроблена наступна таблиця, яка містить повний перелік обладнання.

Таблиця 2.1 – Обране обладнання

| № | Назва | Тип |
|---|---|---|
| 1 | Cisco Catalyst Ws-C3560G-24TS-S | Комутатор |
| 2 | Cisco Catalyst Ws-C2960G-24(48)TC-L | Комутатор |
| № | Назва | Тип |
| 3 | Cisco 2911 | Маршрутизатор |
| 4 | Cisco GLC-LH-SM | Оптичний SFP модуль |
| 5 | Cisco AIR-CAP26021-A-K9 | Безпровідна точка доступу |
| 6 | HP PROLIANT BL660c gen9 | Сервер |

Кінець таблиці 2.1 – Обране обладнання

Проаналізувавши всі наявні дані, отримані в ході розробки ККМ, була створена логічна схема комп'ютерної мережі. Розробка схеми проводилася в векторному графічному редакторі Microsoft Visio.

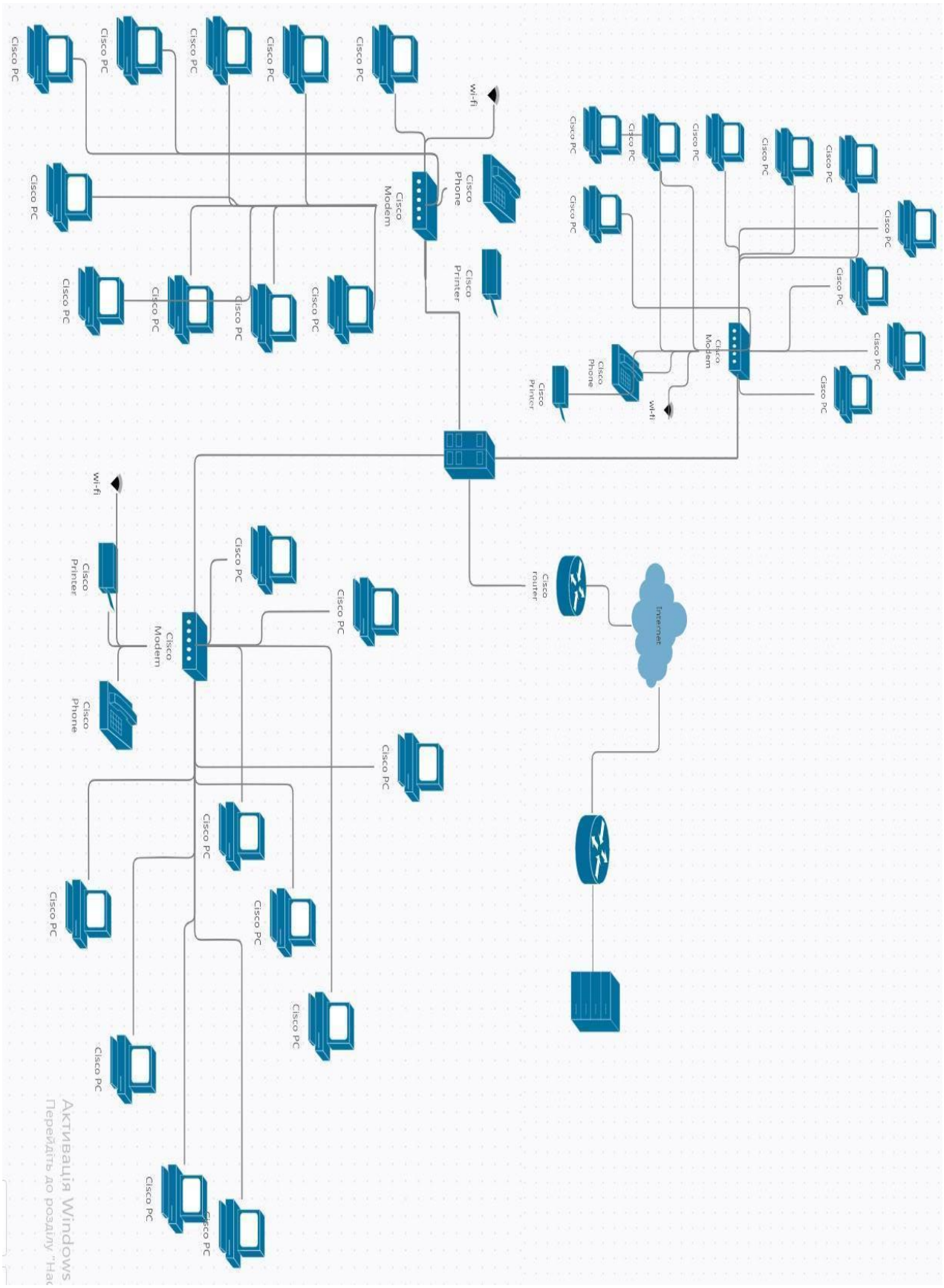



Рисунок 2.12 - Схема мережі

На логічній схемі представлені:

1) фізичні пристрої мережі;
2) типи пристроїв;
3) імена пристроїв;
4) IP-адресація управління пристроями;

## 2.6 Висновки

Представлено перелік важливих критеріїв для дослідницьких проектів у галузі комп’ютерних мережевих технологій та застосування мереж до базових комп’ютерних та інформаційних досліджень. Визначено конкретні області досліджень, що заслуговують на підтримку. Наводяться короткі знімки кожного з цих напрямів досліджень.

# 3 ОПТИМІЗАЦІЯ МУРАШИНИМИ КОЛОНІЯМИ

Оптимізація колонії мурашок (ACO) - парадигма для проектування мета евристичних алгоритмів для комбінаторних задач оптимізації. Перший алгоритм, який можна класифікувати в цих рамках, був представлений в 1991 р. і з тих пір у літературі повідомляється про багато різноманітних варіантів основного принципу. Суттєвою рисою алгоритмів ACO є поєднання апріорної інформації про структуру перспективного рішення з апостеріорною інформацією про структуру раніше отриманих хороших рішень.

Метаевристичні алгоритми - це алгоритми, які, щоб уникнути локальних оптимумів, керують деякими основними евристиками: або конструктивна евристика, починаючи з нульового рішення та додаючи елементи для побудови хорошого повного, або евристика локального пошуку, починаючи з повного рішення та ітеративно модифікуючі деякі його елементи, щоб досягти кращого. Мета Евристична частина дозволяє евристиці низького рівня отримувати рішення кращі, ніж ті, які він міг би досягти самостійно, навіть якщо їх повторити. Зазвичай механізм контролю досягається або обмеженням, або шляхом рандомізації набору рішень локальних сусідів, які слід враховувати при локальному пошуку (як це відбувається у випадку модельованого відпалу або пошуку табу), Характеристикою алгоритмів ACO є явне використання ними елементів попередніх рішень. Насправді вони керуються конструктивним низькорівневим рішенням, як це робить GR MCP, але включаючи його в систему популяцій та рандомізуючи конструкцію способом Монте-Карло. Комбінація Монте-Карло різних елементів рішення пропонується також Генетичними алгоритмами, але у випадку ACO розподіл ймовірності чітко визначається попередньо отриманими складовими рішення. Конкретний спосіб визначення компонентів та пов'язаних з ними ймовірностей залежить від конкретної проблеми і може бути розроблений по-різному, стикаючись із компромісом між специфічністю інформації, яка використовується для кондиціонування, та кількістю рішень, які потрібно створити до ефективно упереджуючи розподіл ймовірностей, сприяючи появі хороших

рішень. Таким чином, різні додатки віддають перевагу використанню умов на рівні змінних прийняття рішень вимагає величезної кількості ітерацій, перш ніж отримати точний розподіл, або обчислювальна ефективність, використовуючи таким чином дуже грубу інформацію про кондиціонування.

## 3.1 Оптимізація колонії мурашок

В роботі пропонується метод … ОПМ - це клас алгоритмів, першим членом якого називається Ant System спочатку запропонований Колорні, Доріго та Маньєццо. Основною ідеєю, яка вільно натхнена поведінкою справжніх мурах, є паралельний пошук над кількома конструктивними обчислювальними потоками на основі локальних даних про проблему та на динамічній структурі пам'яті, що містить інформацію про якість раніше отриманого результату. Колективна поведінка, що виникає в результаті взаємодії різні пошукові потоки довели свою ефективність у вирішенні комбінаторної оптимізації (СО) проблеми.

Слідуючи, ми використовуємо такі позначення. Комбінаторна оптимізація проблема - це проблема, визначена над множиною C = c 1, ..., c n основних компоненти. A підмножина S компонентів являє собою рішення проблеми; F ⊆ 2 C. є підмножиною можливі рішення, таким чином рішення S можливо тоді і лише тоді S ∈ AA функція витрат ції z визначається над областю рішення, z: 2 C. a R, мета полягає в тому, щоб знайти мінімально можливе рішення S *, тобто знайти S *: S * ∈ F i z (S *) ≤ z ( S), ∀S ∈ Ф.

З огляду на це, функціонування алгоритму ОПМ можна узагальнити наступним чином. Набір обчислювальних одночасних та асинхронних агентів (колонія мурах) рухається через стани задачі, що відповідають частковим рішенням розв'язування задачі. Вони рухаються, застосовуючи стохастичне місцеве рішення політика, заснована на двох параметрах, які називаються маршрутами та привабливістю. Переміщаючись, кожен мураха поступово будує рішення проблеми. Коли мураха завершує а розчину, або на етапі будівництва мураха

оцінює розчин і модифікує значення сліду на компонентах, що використовуються в його розчині. Ця інформація про феромони направить пошук майбутніх мурах.

Крім того, алгоритм ОПМ включає ще два механізми: випаровування слідів і, за бажанням, дії демона. Випаровування сліду зменшує всі значення сліду з часом, щоб уникнути необмеженого накопичення стежок за деяким складом. Діяння демона можуть бути використані для реалізації централізованих дій, які не можуть виконуватися одиничними мурахами, наприклад, виклик локальної процедури оптимізації або оновлення загальної інформації, яка буде використана для прийняття рішення про упередження процес пошуку з нелокальної точки зору.

Більш конкретно, мураха - це простий обчислювальний агент, який ітеративно конструює рішення для екземпляра для вирішення. Часткові рішення проблем розглядаються як штатів. В основі алгоритму ОПМ лежить цикл, де на кожній ітерації кожен мураха рухається (виконує крок) зі стану і в інший стан у, що відповідає а більш повне часткове рішення. Тобто на кожному кроці s кожен мураха k обчислює набір Ак s (i) можливого розширення до поточного стану та переходить до одного з них у ймовірність. Розподіл ймовірностей визначається наступним чином. Для мурашки k -ймовірність пій k перехід від стану і до стану у залежить від поєднання два значення:

Привабливість h руху, як обчислюється деякими евристичними показниками апріорна бажаність цього кроку;

Рівень маршруту t ходу, вказуючи, наскільки він був досвідченим у минулому зробити цей конкретний крок: отже, він представляє апостеріорне вказівку бажаності цього кроку.

Траси оновлюються, як правило, коли всі мурахи завершили своє рішення, збільшуючи або зменшуючи рівень стежок, що відповідає ходам, які були частиною "хороші" чи "погані" рішення відповідно.

## 3.2 Особливості мурашиних систем

Важливість оригінальної мурашиної системи (МС) полягає головним чином у тому, що вона є прототипом ряду мурашиних алгоритмів, які спільно реалізують Парадигма ОПМ. МС вже слідує схемі, представленій у попередньому підрозділі, вказуючи її елементи наступним чином.

Розподіл ймовірності переміщення визначає ймовірності p рівними 0 для всі ходи, які неможливо здійснити (тобто вони знаходяться в списку табу ant k, тобто це список містять усі ходи, які неможливі для мурашок k, починаючи зі стану i), інакше вони обчислюються за формулою (3.1), де a і b - параметри, визначені користувачем ($0 £ a, b £ 1$ ):

$$P \frac{K}{I\omega} \{ \frac{T\frac{A}{\omega}+N\frac{\beta}{\psi}}{\sum_{(IC)\notin labuk} (t\frac{a}{IC}+\eta\frac{\beta}{IC})} \tag{3.1}$$

Після кожної ітерації t алгоритму, тобто коли всі мурахи завершують рішення, стежки оновлюються за допомогою формули(3.1)

$$t_{l\psi}(t) = \rho T_{l\psi}(t - l) + \Delta t_{t\Psi} \tag{3.2}$$

де D представляє суму внесків усіх мурах, які використовували рух (i) для побудови їх рішення, ($0 £ a, b £ 1$ ):, - це визначений користувачем параметр коефіцієнт випаровування, а D представляє суму внесків усіх мурахи, які використовували move (i) для побудови свого рішення. Внески мурашок складають пропорційна якості досягнутих рішень, тобто, чим кращим є рішення, тим вище буде додатковий внесок, доданий до ходів, які він використовував. Наприклад, у випадку TSP, переміщення відповідають дугам графіка, таким чином стан і може відповідати шляху, що закінчується вузлом i, стан y - тому ж шлях, але з дугою (ij), доданою в кінці, і переміщення буде обходом дуга (ij). Якість розчину мурашки k буде довжиною L туру знайдений мурашкою і формула (5.2) стане tij з.$tij\ (t) = r\ t\ ij\ (t-1)\ +\ Dt\ Ij$

$$\Delta t_{ij} = \sum_{k=1}^{m} \quad \Delta t \frac{k}{ij}) \tag{3.3}$$

де m - кількість мурах і kij Dt - кількість сліду, прокладеного по краю (ij) за ant k, яке можна обчислити як

$$\Delta t \frac{k}{ij} = \{ \frac{Q}{Lk} \; if \; ant \; k \; uses \; (ij) \; in \; its \; tour \tag{3.4}$$

Q - постійний параметр.

Мурашина система просто повторює головний цикл, де мурахи будують паралельно їх рішення, після чого оновлення рівнів маршруту. Продуктивність алгоритму залежить від правильної настройки декількох параметрів, а саме: a, b, відносного важливість траси та привабливість, r, стійкість траси, tij (0), початковий рівень траси, m, кількість мурашок та Q, що використовуються для визначення високоякісних рішень з низька вартість. Алгоритм такий.

1) {Ініціалізація}
Ініціалізувати і , ()
2) {Бідувництво}
на кожного мураху к (в даний час в штаті і) робити
повторити
вибрати з імовірністю стан, в який потрібно перейти.
додати вибраний хід до к- набір табу мурахи к.
до мурашки к-завершив своє рішення, кінець
для
3) {Оновлення маршруту}
За кожен рух мурахи() робити
обчислити х
оновити матрицю слідів
кінець for
4) {Припинення дії}
Якщо ні(кінцевий тест), перейдіть до кроку 2.

## 3.3 Система колоній мурашок

МС був першим алгоритмом, натхненним поведінкою справжніх мурах. Спочатку застосовували МС до вирішення проблеми мандрівного продавця, але не зміг скласти конкуренцію проти найсучасніших алгоритмів у цій галузі. З іншого боку, він має заслужити впровадження алгоритмів ОПМ та показати потенціал використання штучних феромони та штучні мурахи для стимулювання пошуку завжди кращих рішень складні завдання оптимізації. Наступні дослідження були мотивовані двома цілі: перша полягала в підвищенні продуктивності алгоритму, а друга було розслідувати та краще пояснити його поведінку. Гамбарделла та Доріго запропонували в 1995 р. Алгоритм Ant-Q, розширення МС, яке інтегрує деякі ідеї Q-learning, а в 1996 році система колоній мурашок (СКМ) а спрощена версія Ant-Q, яка підтримувала приблизно однаковий рівень продуктивність, що вимірюється складністю алгоритму та обчислювальними результатами. Оскільки СКМ є основою багатьох алгоритмів, визначених у наступні роки, ми зосередився на цьому увагу на АСУ, крім Ant-Q. САУ відрізняється від попередньої АС тим, що з трьох основних аспектів

## 3.4 Феромон мурашиного алгоритму

У СКМ після того, як усі мурахи обчислили свій тур (тобто в кінці кожної ітерації) МС оновлює феромоновий слід, використовуючи всі розчини, вироблені колонією мурашок. Кожне ребро, що належить одному з обчислювальних рішень, модифікується на величину феромону, пропорційний значенню його розчину. Наприкінці цього етапу феромон всієї системи випаровується і в процесі побудови і оновлення повторюється. Навпаки, в СКМ лише найкраще рішення, обчислене з тих пір початок обчислень використовується для глобального оновлення феромону. Як так було в МС, глобальне оновлення має на меті підвищити привабливість перспективний маршрут, але механізм САУ є більш ефективним, оскільки дозволяє уникнути тривалого часу конвергенції, безпосередньо сконцентрувавши пошук у районі найкращого туру знайдено до поточної ітерації алгоритму. В АСУ остаточна фаза випаровування заміщується локальним

оновленням феромон, застосовуваний на етапі будівництва. Щоразу, коли мураха рухається у поточне місто до наступного феромону, пов'язаного з краєм, модифіковано в наступним чином: tij (t) = r × t ij (t -1) + (1- r) × t 0, где 0 £ r £ 1 – параметр (зазвичай встановлюється на рівні 0,9), а t0 - початкове значення феромону. t0 визначається як t0 = (n · Lnn) -1, де Lnn - довжина туру, отримана при виконанні одного САУ ітерація без феромонного компонента (це еквівалентно імовірнісному найближчий сусід евристичний). Ефект локального оновлення полягає у зростанні бажаності краї динамічно змінюються: кожен раз, коли мураха використовує ребро, це стає дещо менш бажаний і лише для ребер, які ніколи не належали до світового найкращого тур феромон залишається t0. Цікава властивість цих локальних і глобальних Механізми оновлення полягають у тому, що феромон tij (t) кожного краю нижчий обмежений за t0. Подібний підхід був запропонований для Max-Min-MC (MM,) чітко вводить нижню та верхню межі значення феромонів випробувань. Правило переходу держави Під час побудови нового рішення правилом переходу держави є фаза де кожен мураха вирішує, до якого наступного стану переїхати. В САУ новий стан. Введено правило переходу, яке називається псевдовипадково-пропорційним. Правило псевдовипадкової пропорції є компромісом між псевдовипадковим станом правило вибору, яке зазвичай використовується в Q-навчання, і правило вибору випадкової пропорції дії, яке зазвичай використовується в Ant System. З псевдовипадковим правилом обраний стан найкращий з імовірністю q0 (експлуатація), поки є випадковий стан обраний з імовірністю 1-q0 (розвідка). Використання MC випадково-пропорційне правило наступний стан вибирається випадковим чином із розподілом ймовірностей залежно від h і t. Правило псевдовипадково-пропорційного пропорційного стану СКМ забезпечує а прямий спосіб балансу між дослідженням нових станів та використанням апріорних та накопичених знань. Найкращий стан вибирається з імовірністю q (що є параметром 0 £ q0 £ 1, зазвичай фіксованим на 0,9) і з імовірністю (1-q0) наступний стан вибирається випадковим чином із розподілом ймовірностей на основі h і t зважені на a (зазвичай дорівнює 1) і b (зазвичай дорівнює 2).

$$S = \{\frac{arg\ max\ \{Tij\ a \cdot Nij\beta\}\ \ if\ q \leq q0\ (exploitat\ ion)}{As\ rule\ 2/1 \quad otherwise\ \ (explorati\ on)} \tag{3.5}$$

## 3.5 Гібридизація та підвищення продуктивності

АСУ застосовували для вирішення великих симетричних та асиметричних подорожей проблеми продавця (TSP / ATSP). Для цих цілей САУ включає в себе вдосконалена структура даних, відома як список кандидатів. Список кандидатів є статичним структура даних довжини cl, яка містить для даного місця і переважні міста cl до відвідування. Мураха в СКМ спочатку використовує список кандидатів із правилами переходу держави вибрати місто, куди переїхати. Якщо жодне з міст у списку кандидатів не може бути відвідане, мураха обирає найближче місто, лише використовуючи евристичне значення h. СКМ для TSP / ATSP вдосконалено шляхом включення евристики локальної оптимізації (гібридизація): ідея полягає в тому, що кожного разу, коли рішення генерується доводиться до свого локального мінімуму шляхом застосування локальної евристики оптимізації заснована на стратегії обміну країнами, як 2-opt, 3-opt або Lin-Kernighan. нові оптимізовані рішення розглядаються як остаточні рішення, отримані в поточній ітерації мурахами, і використовуються для глобального оновлення феромонних шляхів. Це впровадження АСУ, що поєднує нову політику управління феромонами, а Нова стратегія переходу держави та процедури місцевого пошуку була остаточно конкурентно спроможною з найсучаснішим алгоритмом вирішення проблем TSP / ATSP. Це відкрив нову межу для алгоритму, заснованого на ОПМ. Дотримуючись того самого підходу що поєднує в собі конструктивну фазу, керовану феромоном, та локальний пошук Фазу, яка оптимізує обчислюване рішення, алгоритми ОПМ змогли зламати кілька записів оптимізації, в тому числі для проблем аршрутизації та планування що буде представлено в наступних параграфах.

## 3.6 ANTS

ANTS - це розширення MC, запропоноване в, яке визначає деякі невизначені елементи загального алгоритму, такі як функція привабливості для використання або ініціалізація розподілу слідів. Це виявляється варіацією загальна структура ОПМ, яка робить отриманий алгоритм подібним за структурою алгоритми пошуку по дереву. Насправді, основна риса, яка відрізняє ANTS від а алгоритм пошуку дерева - це відсутність повного механізму зворотного відстеження, що є підмінений імовірнісним (недетермінованим) вибором держави, в яку потрібно перейти та шляхом неповного (Приблизного) дослідження дерева пошуку: це обґрунтування назви ANTS, що є абревіатурою Апроксимаційного недетермінованого пошуку дерева. Далі ми окреслимо два відмінні елементи алгоритму ANTS в рамках ОПМ, а саме функцію привабливості та механізм оновлення слідів.

## 3.7 Механізми оновлення маршруту

Хороший механізм оновлення слідів дозволяє уникнути застою, небажаної ситуації які всі мурахи неодноразово конструюють однакові рішення, унеможливлюючи подальше дослідження в процесі пошуку. Застій виникає через надмірний слід рівня на ходах одного рішення, і може спостерігатися на просунутих фазах процес пошуку, якщо параметри неправильно налаштовані на проблему. Процедура оновлення слідів оцінює кожне рішення порівняно з останніми k рішеннями глобально побудований ANTS. Як тільки доступно k рішень, їх переміщення середнє z обчислюється; кожен новий розчин zcurr порівнюється з z (а потім використовується для обчислення нового значення ковзної середньої). Якщо curr нижчий за z, слід рівень доходів останнього рішення збільшується, інакше він зменшується. Формула (5.6) визначає, як це реалізовано:

$$\triangle Tij = To \cdot (1 - \frac{Zcurr - LB}{z - LB}) \tag{3.6}$$

де z - середнє значення останнього k рішення, а LB - нижня межа на оптимальна вартість вирішення проблеми. Дозволяє використання процедури динамічного масштабування дискримінація невеликого досягнення на останньому етапі пошуку, уникаючи при цьому зосередження пошуку лише навколо гарних досягнень на самих ранніх етапах. Одним з найскладніших аспектів, який слід враховувати в метаевристичних алгоритмах, є компроміс між розвідкою та експлуатацією. Для отримання хороших результатів, агент повинен віддавати перевагу діям, які він намагався в минулому і визнав їх ефективними отримання бажаних рішень (експлуатація); але щоб їх виявити, йому слід спробувати дії, не вибрані раніше (розвідка). Ні розвідки, ні експлуатації можна переслідувати виключно, не виконавши завдання: з цієї причини АНТИ алгоритм інтегрує процедуру уникнення застою для полегшення розвідки з механізмом визначення ймовірності на основі привабливості та слідів для визначення бажаності ходів.

1) **Обчисліть** (лінійну) нижню межу ЛВ до задачі
   Ініціалізувати т(,) з основними значеннями змінних
2) **Для** к=1 м (м = кількість мурах) робити

повторити

2.1) обчислити ()
2.2) вибрати з імовірністю стан, в який потрібно перейти, додати
2.3) вибраний хід до к-список табу мурахи
   до мурашка к-завершив своє рішення
2.4) перенесіть розчин до його локального оптимуму
   кінець for
3.) Для кожен рух мурахи ()
4. ) Якщо ні (кінцевий_тест) крок 2.

Можна відмітити, що загальна структура алгоритму ANTS дуже схожа до стандартної процедури пошуку по дереву. На кожному етапі ми маємо фактично частковий розчин, який розширюється розгалуженням на все можливе потомство; тоді зв’язок – це обчислюється для кожного потомства, можливо, осягнення домінуючого та потоку часткове рішення обирається з числа тих, що пов'язані з пожилими нащадками на основі міркувань нижньої межі. Просто додавши зворотне відстеження та усуваючи вибір MonteCarlo вузла, до якого потрібно перейти, ми повертаємось до стандарту відгалуження та прив’язана процедура. Тому в код ANTS можна легко перетворити точна процедура.

## 3.8 Проблеми оптимізації

Проблеми оптимізації поширені у багатьох сферах та різних сферах людської діяльності, де нам потрібно знайти оптимальні або майже оптимальні рішення для конкретних проблем із можливістю задовольнити деякі обмеження. Більш конкретно, оптимізація фокусується на розробці ефективних та потужних обчислювальних інфраструктур, які, серед іншого, будуть використовуватися з метою пришвидшення мета евристичних методів шляхом значного поліпшення їх продуктивності. Таким чином, було розроблено багато евристичних алгоритмів для пошуку більш швидких, майже оптимальних рішень. Евристичні алгоритми можуть швидко створити рішення з прийнятною якістю. До евристики та мета-евристики належать генетичні алгоритми, алгоритм колонії мурашок, з модельований відпал, оптимізація сірого вовка тощо. У цьому розділі критично розглядаються різні методи метаевристичної оптимізації, що використовуються для оптимізації енергії в бездротовій сенсорній мережі, а також їх детальний аналіз та оцінка на основі різних параметрів. Дослідження та оцінка корисні для покращення ефективності існуючих методів, а також для розробки нових методів.
Далі в цьому розділі ми представимо застосування алгоритмів ОПМ для деякі значні комбінаторні проблеми оптимізації. Це для того, щоб дати читачеві уявлення про те, що пов'язане з використанням алгоритму ОПМ для проблеми: навіть хоча в

останньому підрозділі представлений огляд нещодавньої заявки, за списком жодним чином не є вичерпним, оскільки це стає очевидним при пошуку в Інтернеті під ключові слова “оптимізація колонії мурашок”.

## 3.9 Квадратичне завдання на присвоєння

Квадратична задача присвоєння (КЗП) - це проблема призначення n об'єктів до n місць, щоб мінімізувати вартість призначення, де вартість визначена за допомогою квадратичної функції. КЗП вважається однією з найскладніших проблем СО, і може бути вирішено оптимально лише для невеликих випадків. Кілька додатків ОПМ мали справу з КЗП, починаючи з використання МС, а потім за допомогою декількох з більш досконалих версій. Обмежена ефективність АС була в факт покращився за допомогою добре налаштованого локального оптимізатора, але декількох інших систем раніше введені також були адаптовані до КЗПП. Для тестування алгоритмів рішення КЗП Тейяр запропонував класифікувати екземпляри на чотири групи: (i) неструктуровані, однорідні випадкові (ii) неструктуровані, відстань сітки, (iii) реальний та (iv) реальний. І М МС-КЗПП, і МС-КЗПП мають застосовувались до екземплярів проблем типу і та iii. Виступи цих двох евристичні підходи сильно залежать від типу проблеми. Порівняння з деякими з найкращих евристик для КЗП показали, що МС-КЗПП працює що стосується реальних, нерегулярних та структурованих проблем. На з іншого боку, щодо випадкових, регулярних та неструктурованих проблем результативність цього техніка менш конкурентоспроможна. Ця залежність від проблеми не була показана ANTS, що також застосовувалось до КЗПП. Для того, щоб застосувати ANTS до КЗП (або будь-якої іншої проблеми), необхідно вкажіть нижню межу для використання та те, що є кроком у контексті проблеми (крок 2.2). Додаток, описаний у, зробив наступний вибір. Що стосується нижньої межі, оскільки в даний час для КЗП немає нижньої межі, яка є одночасно жорстким та ефективним для обчислення, використано обмеження LBD, що може бути обчислюється в O (n), але що, на жаль, в середньому досить далеке від оптимального рішення. Що стосується

ходів, було оголошено, що ход відповідає призначенню об'єкта до місця, таким чином додаючи новий компонент до часткового рішення, що відповідає стану, з якого відбувся переїзд. Деякі міркування щодо структуру переміщення використовували для підвищення обчислювальної ефективності результуючого алгоритму. ANTS тестували на екземплярах до n = 40 і показали свою ефективність на всіх типах екземплярів; до того ж його пряме транспонування в точну гілку і пов'язане було також ефективний у порівнянні з іншими точними алгоритмами.

## 3.10 Застосування підходів ОПМ до проблем

У цьому розділі викладено деякі з останніх застосувань підходів ОПМ до проблеми, крім перелічених у попередніх. Цей сорт добре представлений на багатьох різноманітних конференціях із композиціями, повністю присвяченими ОПМ та найвизначніше в серії конференцій ANTS, повністю присвяченій натхненним алгоритмам шляхом спостереження за поведінкою мурах. Було представлено багато різних додатків: від об’єднання планів до маршрутизації проблеми, від планування драйверів до спільного використання простору пошуку, від набору покриттів до планування роботи медсестри, від забарвлення графіків до динамічного збалансування кількох критеріїв проблеми. Значна частина відповідної літератури може бути доступна в Інтернеті. Більше того, було опубліковано кілька вступних оглядів. Серед проблем, яких немає у списку вище, чільну роль відіграє TSP. Насправді, TSP був і, у багатьох випадках, досі є першим тестом для ОПМ варіанти, і більше загалом для більшості метахевристик комбінаторної оптимізації. Вже з цією проблемою з'явилася обмежена ефективність перших варіантів, і це сприяло розробці удосконалення підходів, що модифікують якийсь елемент алгоритму і, можливо, гібридизуючи фреймворк з жадібним локальним пошук або з іншими підходами, такими як генетичні алгоритми або табу-пошуки. Потім ці варіанти були застосовані до інших проблем, наприклад, до прикладу MAXMIN мурашина система була застосована до проблеми потоку магазину в, проблема тоді стикався також з іншими

модифікаціями ОПМ , тоді як у описаний підхід до TSP на основі рангу або в так званий найкращий-найгірший варіант. Зовсім недавно різні автори вирішували питання TSP гібридні варіанти, переважно з використанням пошуку за допомогою табу, але також, у випадку великих випадків TSP, також з генетичною еволюцією та пошуком найближчих сусідів, з метою підвищення як ефективності, так і ефективності. Більше того, варіанти базового TSP, такі як проблема орієнтування або імовірнісний TSP, де повинні бути клієнти також відвідали з певною ймовірністю. Проблеми з плануванням є ще однією спільною областю для перевірки ефективності алгоритмів ОПМ. Підхід ОПМ для планування роботи в магазині представлений у , тоді як заявки на реальні справи щодо планування. Зовсім недавно про зрілість галузі свідчить той факт, що підходи ОПМ почали пропонувати також для проблем, які не є стандартною комбінаторною оптимізацією, але більш безпосередньо пов'язані з реальною практикою. Наприклад, проблема пошуку та кластеризації записів великих розмірів бази даних стикаються за допомогою ОПМ, тоді як алгоритм для документа кластеризація Ще більше теоретичних проблем пов'язано з просторовими аналіз даних розглядався методами ОПМ. Нарешті, недавня цікава наукова галузь АКО, безпосередньо не пов’язана з комбінаторна оптимізація, стосується телекомунікацій. Насправді площа пакета комутаційні комунікації представляються перспективним полем для маршрутизації, пов'язаної з ОПМ підходи.

## 3.11 Збіжності алгоритмів мурашиних колоній

Нещодавно з’явилися деякі роботи, які дають теоретичне уявлення про властивості збіжності алгоритмів колонії мурашок. Усі докази стосуються спрощених версій фактично використовуваних систем і не містять прямих вказівок для реального використання, але вони представляють інтерес для з'ясування загальних властивостей використовуваних систем. Перші такі докази запропонував Гутяхр, який працював над ОПМ варіант, який називається Graph-BMC Ant System (CBMC). Назва походить від аналіз проводиться на так званому конструктивному

графіку, який є графіком, присвоєним екземпляру розглянутої задачі оптимізації, кодує можливих рішень шляхом "прогулянок" по графіку. Значення цільової функції ходьба дорівнює цільовій функції значення відповідного можливого рішення вихідної проблеми. Завжди можна розробити графік побудови для будь-який заданий екземпляр задачі комбінаторної оптимізації з кількістю вузлів лінійна за кількістю бітів, необхідних для подання рішення, і кількість дуг, квадратних у цій кількості бітів. Гутяхр довів, що за перелічених нижче умов рішення, що генеруються в кожній ітерації даного Графіка Мурашина система сходиться з імовірністю, яку можна довільно наблизити до 1 до оптимальне рішення даного екземпляру задачі. Основними умовами є: (i) існує лише одна оптимальна прогулянка в Ш, тобто оптимальне рішення є унікальним, і воно є кодується лише однією прогулянкою у W; (ii) уздовж оптимальної прогулянки w * - бажаність значення задовольняють hkl. (u)> 0 для всіх дуг (k, l) w * та відповідної часткової ходить u з w *; (iii) використовується версія того, що називається елітарною стратегією, де тільки найкращі прогулянки винагороджуються: прогулянки, в яких переважає інша вже помилкові прогулянки більше не отримують приросту феромону. Особливо перший з них умови досить обмежувальні. Штуцер та Доріг пропонують ще один доказ збіжності. Це властивість, яка вже гарантована лише випадковим пошуком, і є не загубитися, накладаючи мінімальне значення сліду. Більше того, автори показують, що це так можна обчислити нижню межу ймовірності найкращого поточного рішення бути оптимальним. Нарешті, Gutjahr в недавній роботі спирається на ці результати контексту ОПМ і показує, що для певного алгоритму ОПМ модифікація, яка залежить від часу СВМС, поточні рішення сходяться до оптимального рішення з імовірністю рівно одному. Більш конкретно, він показує, що використовуючи відповідні схеми параметрів, він можна гарантувати, що оптимальні шляхи отримують атрактори стохастичної динаміки процес, реалізований алгоритмом. Це покращує всі попередні результати і доводить а властивість тієї ж міцності, що і наймічніша, до цього часу отримана в цілому мета евристична область, яка була отримана Хаєком для модельованого відпалу.

## 3.12 Висновки

Планування шляху є ключовим питанням у галузі багатьох досліджень . Його основна мета - знайти оптимальний або неоптимальний, безпечний і без зіткнень шлях від вихідної точки до цільової точки в навколишньому середовищі з перешкодою. За ступенем інтелекту в процесі планування шляху планування шляху для мереж можна розділити на традиційне планування шляху та інтелектуальне планування шляху. Однак традиційні методи неможливо додатково вдосконалити з точки зору ефективності пошуку шляхів та оптимізації шляхів.

# 4 ЕФЕКТИВНІСТЬ РОЗПОДІЛЕНИХ СИСТЕМ

Існує дві основні концепції, які обмежують ефективність розподіленої системи:

Затримка час, необхідний для передачі повідомлення з одного місця в інше в розподіленій системі.

Пропускна здатність : кількість даних, яка може бути передана за одиницю часу в стабільному стані.

Крім цього, тут слід включити два провідні показники ефективності:

Час відповіді : час, необхідний для отримання результату після подання завдання на обробку системою.

Пропускна здатність : здатність системи витримувати великі навантаження. Іншими словами, кількість виконаних завдань за одиницю часу.

Метою розробленої розподіленої обчислювальної системи є досягнення результатів балансування навантаження, іншими словами, мінімізація затримки та часу відгуку при одночасному збільшенні пропускної здатності простими словами поліпшити продуктивність PC . Для цього необхідно здійснити певні оптимізації системи.

Балансування навантаження, щоб отримати максимальну продуктивність, кожному апарату в системі потрібно призначити однаковий обсяг роботи. Оскільки різні завдання не обов’язково вимагають однакової кількості обчислень, роботи чи часу, балансування навантаження не означає просто прохання кожної машини виконати іменну кількість завдань. Натомість існує два різні підходи до збалансування навантаження обчислювальної системи. Якщо різні розміри завдань відомі заздалегідь, навантаження можна статично збалансувати під час компіляції; якщо, однак, різний розмір завдання невідомий, завдання призначаються різним процесорам динамічно під час виконання.

Паралельно всі машини, які виконують різні частини будь-яких обчислень, повинні для оптимальної продуктивності виконувати різні обчислення одночасно. Це просто оптимізація, заснована на здоровому глузді, якщо є два процесори, які

працюють разом, щоб виконати якийсь великий розрахунок, який потрібно об’єднати, а потім ще раз обробити, третій аналіз вимагає завершення перших двох обчислень перед початком. Таким чином, якщо дві обчислювальні машини працюють одночасно, час, витрачений на очікування закінчення обчислення одним процесором, тоді як інший сидить склавши руки, чекаючи початку нового завдання, буде мінімізований.

Накладні витрати, щоб підвищити ефективність обчислювальної системи, слід мінімізувати обсяг додаткової роботи, яка не потрібна для обчислень. Кожного разу, коли в дану обчислювальну систему вводиться нова змінна, буде застосовано цю нову функцію до старої системи, а не просто запущено нову систему з функціональністю, унікальною для потреб цієї єдиної змінної. У випадку розподілених обчислень, одним із прикладів є необхідність машин взаємодіяти між собою через мережу для обміну результатами. Таким чином, для того, щоб збільшити швидкість доданих процесорів та обчислювальних ресурсів, буде витрачений час на обчислення. Якщо, однак, цей час близький до нуля, накладні витрати були мінімізовані, і час реакції системи значно зменшилася.

Незважаючи на спроби розробників розподілених обчислювальних архітектур підвищити ефективність існуючих систем, існують невід'ємні обмеження щодо вдосконалень, які можна зробити у трьох перелічених вище областях.

Балансування навантаження суттєво обмежене деталізацією завдань, які потрібно виконати. Для багатьох додатків існує обмеження кількості разів на те, що дане завдання може бути розділене; або, іншими словами, є момент, коли завдання повинно оброблятися як завершена сутність, оскільки різні етапи в рамках завдання взаємозалежні. У тому випадку, коли завдання є курсовими (їх не можна розділити багато разів до досягнення атомного рівня), стає важче збалансувати навантаження між різними процесорами. Завдання може суттєво відрізнятися за розміром, але більші завдання не можна розкласти, і, отже, навантаження не може бути рівномірно розподілено між різними процесорами в системі.

При використанні розподіленої обчислювальної системи, де всі обчислення просто виконуються, коли користувач не використовує машину, неможливо мати повністю одночасну систему. Це просто питання практичності, немає можливості отримати оптимальний стан, коли всі обчислення повністю синхронізовані між усіма комп'ютерами в розподіленій системі.

Нарешті, у розподіленій обчислювальній системі завжди будуть накладні витрати на систему. Різні машини повинні розмовляти між собою, і в результаті накладні витрати стають функцією затримки та пропускної здатності системи. Крім того, різним процесорам, можливо, доведеться повторити деякі обчислення, виконані іншими машинами, локально, просто для виконання своїх непомітних завдань. Якщо ці різні завдання виконувались на одній машині послідовно, ці повторювані обчислення могли не знадобитися.

## 4.1 Постановка експерименту

Оптимізація колонії мурашок (ACO) забезпечує інструмент мета евристичної оптимізації та модель колективного інтелекту для кількох додатків, таких як маршрутизація та балансування навантаження. У літературі знайдено багато робіт з використання ACO в балансуванні навантаження. Однак, наскільки мені відомо, не було роботи щодо балансування навантаження в розподілених системах з ACO.

У даній магістерській роботі буде представлений експеримент з ACO для балансування навантаження в розподілених системах. Цей експеримент повністю розподілений, в якому інформація динамічно оновлюється при кожному русі мурахи. Буде прийнята парадигма кількох колоній, така що кожен вузол буде посилати кольорову колонію по всій мережі.

У цьому дослідженні кольорові колонії мурашок використовуються для запобігання руху мурах одного гнізда за тим самим маршрутом і, отже, примусового їх розподілу по всіх вузлах системи, і кожен мураха діє як мобільний агент, який несе нещодавно оновлену інформацію про балансування навантаження до наступного відвіданого вузла. Висновок: Нарешті, ефективність

запропонованого алгоритму ACO порівнюється з підходом викрадення роботи для балансування навантаження в розподілених системах.

## 4.2 Аналіз розподіленої системи балансування

Розподілена система балансування навантаження все ще залишається активною областю досліджень, в якій балансир навантаження намагається поліпшити продуктивність розподіленої системи за допомогою обчислювальної потужності всієї системи для згладжування періодів високих перевантажень в окремих вузлах (Ferrari and Zhou, 1987; Чжоу і Феррарі, 1987), це робиться шляхом перенесення частини робочого навантаження сильно завантажених вузлів на інші вузли для обробки. Рішення щодо того, як збалансувати навантаження між вузлами, є або статичними (Rao та ін., 1979; Тантаві і Талі, 1985; Пінтер і Вольтшталь, 1987; Chen and Shin, 1990) або динамічний (Ni та ін., 1985; Бажання та ін., 1986; Шин і Чанг, 1989; Сю і Хван, 1993; Алі, 2000; 2001). Статичне рішення не залежить від поточного стану системи. Балансування статичного навантаження також можна розглядати як детермінований розподіл завдань у системі, де перевантажений вузол з певною ймовірністю передає деякі свої завдання іншому вузлу, який не залежить від поточного стану системи. Хоча статичне балансування навантаження просто і легко аналізувати за допомогою моделей масового обслуговування, але його потенційна користь обмежена, оскільки воно не пристосовується до стану системи, що змінюється в часі (Eager та ін., 1986).

З іншого боку, динамічне рішення залежить від стану системи на момент прийняття рішення. Коли використовується динамічне вирівнювання навантаження, перевантажений вузол може передати свої завдання іншим вузлам, використовуючи інформацію про поточний стан системи. Динамічна політика за своєю суттю є більш складною, ніж будь-яка статична політика, оскільки вона вимагає, щоб кожен вузол повинен знати стани інших вузлів. Алгоритми балансування навантаження додатково поділяються на кілька кластерів відповідно до обсягу необхідної для них інформації (Shin and Chang, 1989)

Штучний інтелект роїв, зокрема оптимізація колонії мурашок (ACO), є відносно новою обчислювальною та поведінковою парадигмою для вирішення задач оптимізації та комбінації, а отже, її можна використовувати для балансування навантаження; він базується на принципах, що контролюють поведінку природних систем. У такій імітаційній моделі багато розподілених агентів еволюціонують та взаємодіють між собою з метою досягнення глобальної мети, наприклад колоній мурах та зграй птахів. Цей підхід підкреслює розподілену структуру проблеми, прямі або непрямі взаємодії між відносно простими агентами, а також між агентами та їх оточенням.

Застосування ройового інтелекту до проблем мереж виникає, коли група автономних програм (агентів) працює разом. Це називається ACO «Оптимізація колонії мурашок» або мультиагентна система. Кожну особу, програму або автономний модуль можна представити як агента, ці мультиагенти можна використовувати для мережевих додатків, таких як пошук найкоротшого шляху, маршрутизація, балансування навантаження та управління тощо.

Деякі пов’язані роботи з використанням ACO в балансуванні навантаження - це використання мультиагентної системи, два алгоритми запропоновано (Heusse та ін., 1998): Перший заснований на агентах маршрутизації в обидва кінці, які оновлюють таблиці маршрутів шляхом зворотного відстеження шляху після досягнення пункту призначення. Другий спирається на прямих агентів, які оновлюють таблиці маршрутизації безпосередньо, коли вони рухаються до місця призначення. З іншого боку, Салехі та Доллары (2006) представляють систему відлуння розумних, автономних та кооперативних мурах. Мурахи в цьому середовищі можуть розмножуватися, а також можуть покінчити життя самогубством залежно від існуючих умов. Представлена нова концепція, яка називається балансуванням навантаження на рівні Мурашки, для покращення продуктивності механізму. Sim and Sun (2003a) представив підхід багаторазової оптимізації колоній мурашок (MACO) для балансування навантаження в мережах з комутацією каналів. MACO використовує кілька колоній мурашок для пошуку альтернатив оптимальному шляху. Одним із імпульсів MACO є оптимізація роботи

перевантаженої мережі шляхом маршрутизації дзвінків за кількома альтернативними шляхами, щоб запобігти можливим перевантаженням по оптимальному шляху. У MACO кожна група мобільних агентів відповідає колонії мурах, а таблиця маршрутів кожної групи відповідає таблиці феромонів кожної колонії (Sim and Sun, 2003b). Застосовуючи підхід MACO, можливо, можна зменшити ймовірність того, що всі мобільні агенти встановлюють зв’язки, використовуючи лише оптимальний шлях (Sim and Sun, 2003b). Перевага використання MACO в маршрутизований мережі з комутацією каналів полягає в тому, що існує більша ймовірність встановлення з'єднань через кілька шляхів, щоб допомогти збалансувати навантаження, але не збільшує накладні витрати на маршрутизацію (Sim and Sun, 2003b). Буланса кожна група мобільних агентів відповідає колонії мурах, а таблиця маршрутів кожної групи відповідає таблиці феромонів кожної колонії (Sim and Sun, 2003b). Застосовуючи підхід MACO, можливо, можна зменшити ймовірність того, що всі мобільні агенти встановлюють зв’язки, використовуючи лише оптимальний шлях (Sim and Sun, 2003b). Перевага використання MACO в маршрутизації з комутацією каналів полягає в тому, що, швидше за все, встановлюється з'єднання через кілька шляхів, щоб допомогти збалансувати навантаження, але не збільшує накладні витрати на маршрутизацію (Sim and Sun, 2003b).

Застосовуючи підхід MACO, можливо, можна зменшити ймовірність того, що всі мобільні агенти встановлюють зв’язки, використовуючи лише оптимальний шлях (Sim and Sun, 2003b). Перевага використання MACO у маршрутизований мережі з комутацією каналів полягає в тому, що вона, швидше за все, встановлює з'єднання через кілька шляхів, щоб допомогти збалансувати навантаження, але не збільшує накладні витрати на маршрутизацію (Sim and Sun, 2003b). Булансе та ін. (1996), використовували мобільні агенти для інкапсуляції завдань у розподіленій пам'яті, що передають повідомлення. Для того, щоб збільшити гнучкість системи та збалансувати довільне обчислювальне середовище топології графів, у запропонованій моделі агенти мають можливість досліджувати, вивчати та обмінюватися інформацією про навантаження. Запропонована модель була

зрівняна з детермінованими DOSUD та еволюційними алгоритмами. Розподіл запропонованого алгоритму агента на фізичних процесах, як правило, був кращим, ніж розподіл алгоритму DOSUD, коли вимога до часу збільшується.

## 4.2. Матеріали і методи мурашиних колоній

Оптимізація колонії мурашок: Розумна поведінка групи характеризує всю колонію соціальних комах, таких як мурахи; приклади такої поведінки, що виникає, включають видобуток їжі та будівництво гнізд. Цю колективну поведінку можна розглядати як потужну систему вирішення проблем, яка може вирішити такі проблеми, як балансування навантаження. Починаючи з простих взаємодіючих агентів з правилами взаємодії між особами та між індивідами та навколишнім середовищем, мурашина колонія може надати інтелект далеко від будь-яких індивідуальних можливостей. Властивості, пов'язані з їх груповою поведінкою, такі як самоорганізація, гнучкість і стійкість, можуть бути показані як характеристики, які повинні існувати в складній системі управління, оптимізації та техніки вирішення проблем. Проблема того, як майже сліпі мурахи можуть співпрацювати, щоб знайти найкоротший шлях до джерела їжі, привертає етологів протягом декількох років. Було виявлено, що вони спілкуються, прокладаючи феромонні стежки, хімічну речовину, яка приваблює інших мурах, разом із їх рухами до джерела їжі (Доріго та ін., 1996). Мураха воліє з великою ймовірністю йти шляхом, що містить більше феромону, тим самим забезпечуючи слід своїм власним феромоном. Ця колективна поведінка називається автокаталітичною поведінкою, в якій вона визначається як процес позитивного зворотного зв'язку, в якому процес посилюється для підвищення своєї продуктивності; цей зворотний зв'язок примушує процес до швидкого зближення до кінцевого рішення (Доріго та ін., 1996).

Мурахи не добувають їжу лише для пошуку їжі, а скоріше, щоб блукати, шукати, повертатися додому, притягувати до цілі, простежувати феромон і нести їжу (Доріго та Ді Каро, 1999; Доріго, 2001). Феромони також з часом

випаровуються. Як наслідок, феромон стане менш помітним через деякий час, а довші стежки стануть менш привабливими для інших мурах (Tarasevich and Patrick, 2002). Мурахи можуть побудувати найкоротший шлях від свого гнізда до джерела їжі, використовуючи феромонні стежки. Мурашка залишає деяка кількість феромону на землі під час прогулянки. Наступний відчує це і, виходячи з ймовірності, пропорційної кількості феромону, він вибере свій шлях.

Однак штучні мурахи мають деякі основні відмінності від справжніх: По-перше, мурахи мають пам’ять. По-друге, мурахи не є повністю сліпими, нарешті, час штучних мурах є дискретним (Krohn, 2001). Як правило, оптимізація колонії мурашок (ACO) - це евристичний алгоритм загального призначення, який може бути використаний для вирішення різних комбінаторних задач оптимізації (Dorigo et al., 1996). В ACO пошукова діяльність розподілена на штучних мурах, які імітують поведінку справжніх мурах. Перевагами цієї системи є позитивний зворотний зв'язок, розподілене обчислення та використання конструктивної жадібної евристики. Позитивні відгуки стосуються здатності до швидкого відкриття хороших рішень. ACO - це також підхід, заснований на популяції, при якому можна легко досягти паралізації (Dorigo et al., 1996).

Однією з основних важливих концепцій, яку додає ACO, є те, що пошук рішення - це процес, що виникає у процесі співпраці та взаємодії простих агентів. Іншою концепцією є використання непрямого стигмегетичного спілкування шляхом зміни середовища.

Алгоритми мурашок можна розглядати як мультиагентні системи, що використовують штучну стигмергію як засіб для координації штучних мурах для вирішення обчислювальних задач

Колективна діяльність соціальних комах самоорганізується, це означає, що складна групова поведінка виникає внаслідок простих взаємодій індивідів. Результати самоорганізації мають глобальний характер, але в основному походять від місцевої інформації та взаємодії (Tarasevich and Patrick, 2002). Взаємодія комах у суспільстві може мати одну з двох форм: пряму та непряму. Прямі взаємодії можуть мати форму тілесного контакту, зорового контакту та обміну їжею.

Непряма взаємодія також важлива, вона може відбуватися, коли агенти обмінюються інформацією через середовище, в якому вони існують. Таким чином, зберігання інформації відбувається як на рівні колонії, так і на індивідуальному рівні. Це співробітництво шляхом модифікації називається стигмергією (Krohn, 2001).

## 4.3 Основні операції систем оптимізації колоній мурашок

У цій частині виявлено на мою думку усі основні та необхідні операції, які повинні існувати в будь - якій системі ACO, включаючи запропонований експеримент. В ACO мурахи є адаптивними, тобто, якщо середовище зміниться, мурахи шукатимуть кращого рішення. ACO підходить для дискретних задач оптимізації. Основними характеристиками системи змінного струму є позитивні відгуки, розподілене обчислення та конструктивна жадібна евристика (Доріго та ін., 1996). У наступному; буде дано короткий опис основних операцій, задіяних в ACO: Стигмергия непрямий зв'язок через феромони є прикладом позитивного зворотного зв'язку, який називається автокаталітичною поведінкою. І навпаки, негативний зворотний зв'язок виникає через випаровування феромонних слідів (Krohn, 2001).Характеристиками ройової інтелектуальної моделі ACO є нові додані концепції самоорганізації та стигмергії. У розподіленій системі, такій як Ant System, спілкування між агентами має велике значення. Форма спілкування є опосередкованою. Це спілкування можна розглядати як космічну деформацію системи, в якій мешкають мурахи (Крон, 2001). Поведінка всієї колонії мурашок є високо структурованими, взаємодія базується на дуже простих потоках інформації (Колорні та ін., 1992). Загальна ідея алгоритму ACO полягає в тому, що дві протилежні сили застосовують метод оптимізації для досягнення рішення.

Першою силою є автокаталітичний процес, який призводить до появи хороших рішень. Інша сила - це жадібна сила, яка завжди обирає перший найкоротший шлях для пошуку рішення. Жодна з двох сил не може досягти оптимального рішення самостійно.

Але коли вони працюють разом, здається, що жадібна сила може дати правильні поради автокаталітичний силі і дозволити їй дуже швидко сходитися до оптимального значення (Колорні та ін., 1992).

Однак однією з проблем ACO є те, що при значних змінах в навколишньому середовищі потрібно кілька часу, перш ніж мурахи виявляють це і змінюють свою інформацію.Оцінка феромону  Феромон, який відкладає кожен мураха, приваблює наступних мурах, так що вони, ймовірно, будуть шукати в одній області пошукового простору. Взагалі, передбачається, що оцінка феромонів здійснюється місцево мурахами.

Меркле та ін.  запропонував розширити місцевий погляд на мурах за допомогою перспективної стратегії. Меркле та ін. показав, що поведінка Мурашиного алгоритму можна вдосконалити науково, коли мурахи використовують більше інформації, ніж просто місцеві інформаційні значення для свого рішення.

## 4.4 Обмін інформацією при оптимізації колонії мурашок

Обмін інформацією між колоніями мурашок є одним з найважливіших факторів, що впливають на поведінку оптимізації.

Було показано, що кілька колоній можуть діяти дуже ефективно при невеликому обміні інформацією (Міддендорф та ін.). Покращення досягається додаванням трохи обміну інформацією. Було також показано, що обмінюються лише найкращими локальними рішеннями, а не часто достатньо, щоб мати дуже хороші результати.

У (Middendorf et al., 2000) досліджуються різні політики обміну інформацією. Було показано, що обмін невеликою кількістю інформації суттєво впливає на ефективність оптимізації продуктивність. Це можна зробити, дозволивши іншій колонії скористатися хорошими рішеннями, отриманими іншими колоніями.

## 4.5 Запропонована стратегія

У даному експерименті стратегія балансування навантаження залежить від часу та відповідає природній динаміці феромону в реальному житті. У визначений проміжок часу кожен вузол буде діяти як гніздо і посилати кількість мурах (кількість мурах залежить від стану завантаження кожного вузла; перевантажені та недозавантажені вузли відправили більше мурах). Кожен мураха буде подорожувати туром (тривалість туру залежить від розміру системи та стану завантаження вузла). Нарешті, запропонований експеримент буде порівняно зі стандартним алгоритмом викрадення роботи. Розроблена модель повністю розподілена, тобто кожен вузол вузла поводиться незалежно, а також кожен мураха чи агент, це означає, що кожен вузол або мураха є автономними. Рисунок 4.1 представляє додається інформація до кожного вузла або мурахи:

| Вузол ID | Статус завантаження | Феромон |
|---|---|---|

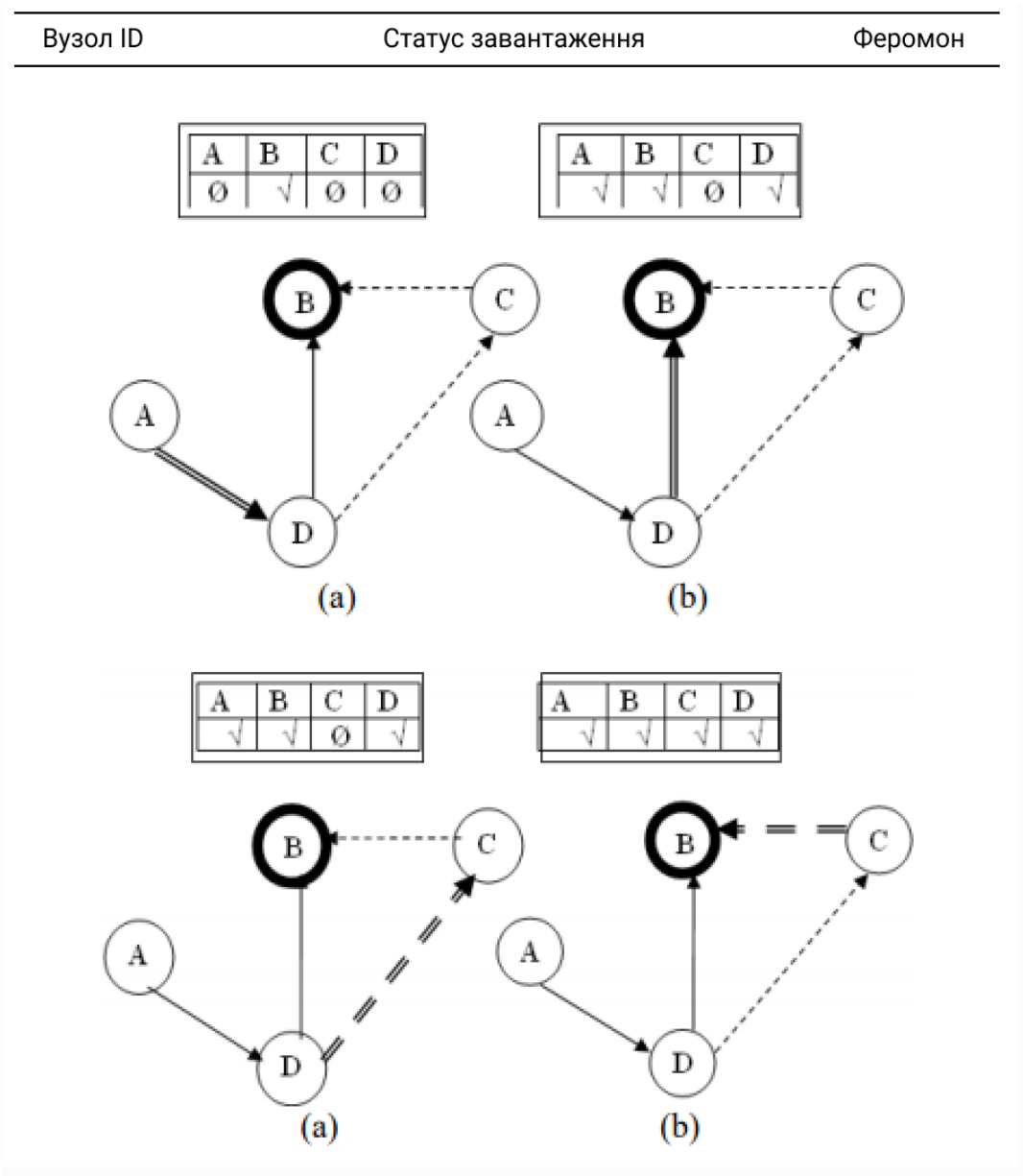



Рисунок 4.1 - Приклад роботи моделі. (а) мураха рухається від А до Г; (б) Мураха рухається з Від D до В; (в) інший мураха рухається від D до С; (d) Мураха рухається від С до В

У представленій моделі кожен вузол містить інформацію про інші вузли в системі (рис. 4.1). У початковому стані записи в таблиці мають значення Null.

У кожному мурашиному турі мураха буде нести оновлену інформацію про всі вузли, через які мураха пройшов. Після прибуття мурахи на кожен вузол будуть виконані наступні дії:

Якщо вузол не має інформації, що міститься в мурашиній таблиці, ця

інформація буде передана до таблиці вузлів без будь-якого оновлення.

Якщо вузол містить інформацію, яка не існує в таблиці мурашок, таблиця мурашок буде оновлена.

Якщо обидва вони мають однакову інформацію, нещодавно оновлена замінить іншу.

## 4.6 Результати

Експеримент був змодельована та протестований, передбачалось, що кількість вузлів у розподіленій системі становить 30, передбачається, що мураха рухається від одного вузла до іншого за 1 часовий крок, кожне завдання передбачає 40 кроків. Для того, щоб підкреслити ефективність запропонованого алгоритму, був розглянутий випадок, коли розподілена система є дуже нерегулярною; передбачається, що вузол № 1 зайнятий 60 завданнями, а інші вузли простоюють. Рис. 4.2 показує ефективність як підходу викрадення роботи, так і підходу колонії мурах. Зрозуміло, що ефективність підходу мурашиних колоній походить від здатності розподіляти інформацію про завантаження по всіх вузлах, тур мурах був обраний випадковим чином, однак розумність мурах щодо перенесення нового стану завантаження до кожного вузла збільшується шанс кожного вузла швидко знайти хороше джерело їжі або зайнятий вузол.

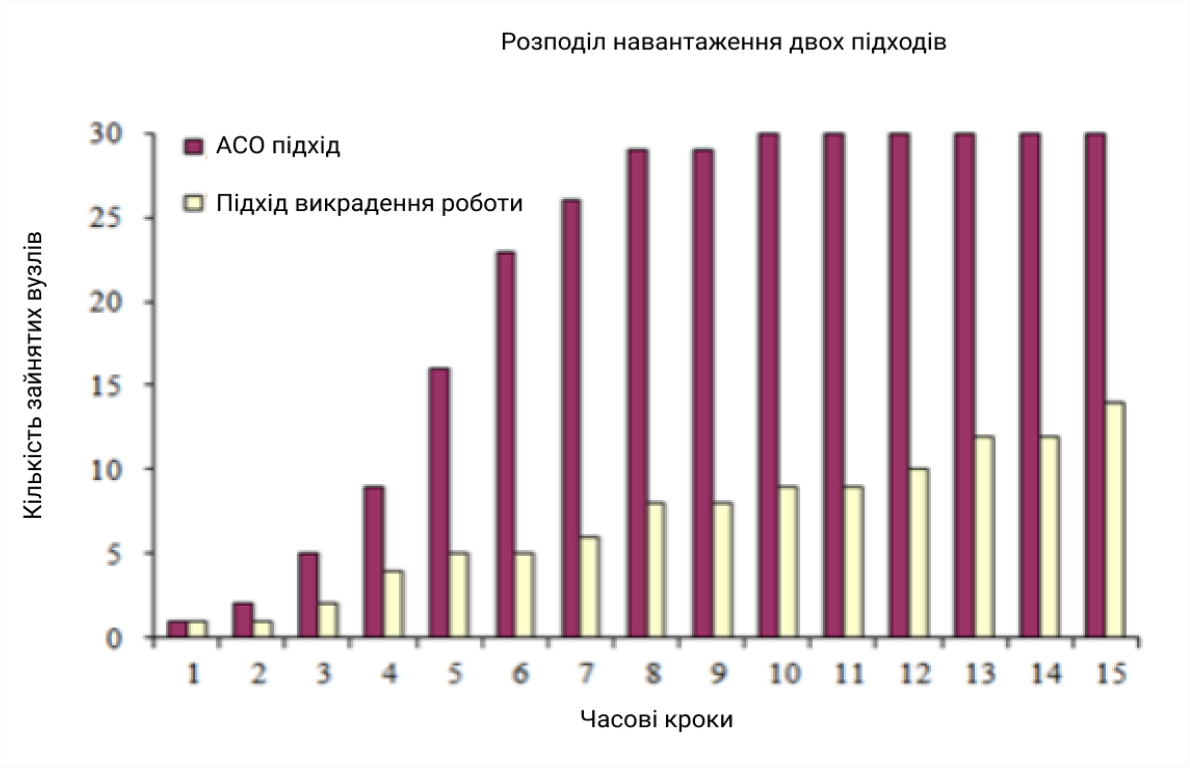



Рисунок 4.2 - Розподіл навантаження кількість зайнятих вузлів у часових кроках

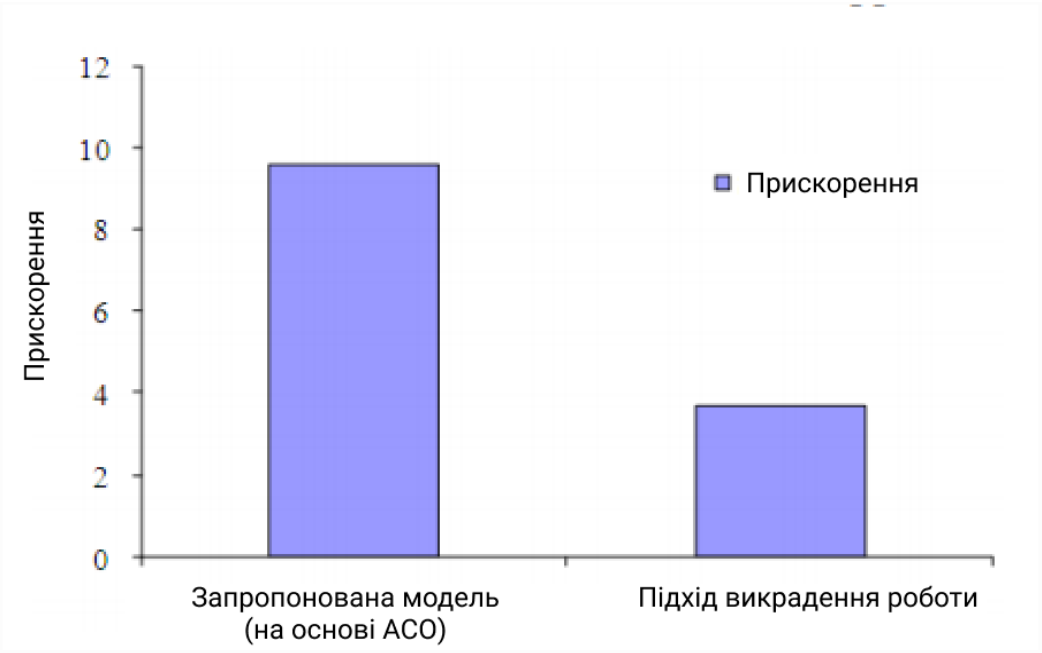



Рисунок 4.3 - Порівняння прискорення запропонованої моделі та моделі викрадення роботи

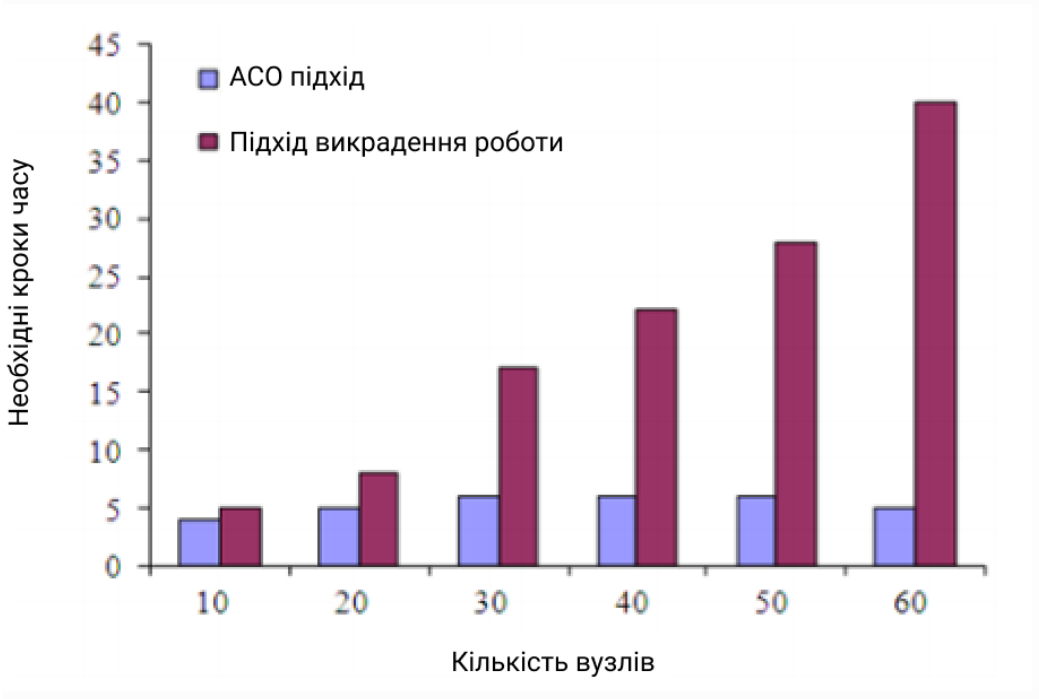



Рисунок 4.4 - Динаміка балансування навантаження: Необхідні кроки часу, необхідні для досягнення розподілу навантаження 50% у порівнянні з кількістю вузлів

Рисунок 4.4 порівнює прискорення запропонованої моделі зі стандартним підходом викрадення роботи. Виходячи з вищевказаних налаштувань, запропонована модель досягла швидкості 9,6, тоді як пришвидшення підходу - 3,7. Рисунок 16 вивчає ефект збільшення кількості вузлів проти кількості кроків, необхідних для підвищення розподілу навантаження до 50%. Показано, що із збільшенням кількості вузлів тимчасові кроки, необхідні для підходу до викрадення робіт, різко зростають, тоді як пропонована модель залишається майже постійною. Цей підхід та його результати дають приклад, який підкреслює важливість ройової системи в процесі прийняття рішень загалом, коли кожен агент може грати не значну роль, а глобальна поведінка може бути надійною та надійною.

Опис основних функцій програмного забезпечення подано в зведеній таблиці

Таблиця 4.1 – Опис основних функцій ПЗ

| № п.п | Назва функції | Опис функції |
|---|---|---|
| 1 | MainWindow(QWidget *parent) | Функція створює основне вікно ПЗ – форму, є базовим об’єктом, який породжує решу елементів ПЗ |
| 2 | setupUi(this) | Функція здійснює налаштування створеного вікна |
| 3 | on_pushButton_5_clicked() | Функція здійснює занесення в список однієї вибраної з мережевих служб |
| 4 | QListWidgetItem() | Функція створює елемент-віджет на формі – список вибору параметрів ланцюжка |
| 5 | listWidget() | Функція викликається для передачі керування функції, яка відповідає за вибраний елемент зі списку |
| 6 | addItem(str) | Функція додає елемент (пукт меню, рядок, змінну) |
| 7 | item(i) | Елемент (пукт меню, рядок, змінна) |
| 8 | on_pushButton_6_ clicked() | Функція здійснює занесення в список усіх мережевих служб |
| 9 | WBoxLayout() | Функція здійснює вивід групованих елементів графічного інтерфейсу |

Продовження таблиці 4.1 – Опис основних функцій ПЗ

| № п.п | Назва функції | Опис функції |
|---|---|---|
| 11 | QListWidgetItem() | Функція створює елементи на формі у вигляді списку |
| 12 | on_pushButton_7_ clicked() | Функція здійснює видалення зі списку однієї вибраної з мережевих служб |
| 14 | removeItemWidget() | Функція видалення рядка таблиці |
| 15 | on_pushButton_8_ clicked() | Функція здійснює видалення зі списку усіх мережевих служб |
| 16 | on_pushButton_clicked() | Функція здійснює виведення інформаційного блоку про ПЗ |
| 17 | statusBar() | Функція реалізує створення елемента графічного інтерфейсу – статус-бар, в якому відображається статур роботи фаєрволу |
| 18 | showMessage() | Функція виведення системної інформації ПЗ про успішне додавання чи видалення правила |
| 19 | on_pushButton_11_clicke d() | Функція виконує системну команду запуску демону (служби в ОС Linux) iptbtes |
| 20 | QString comand() | Функція передає системну команду в консоль операційної системи |
| 21 | comboBox | Функція створює випадаюче меню вибору елементів налаштування фаєрволу (тип ланцюжка, тип таблиці, тип протоколу) |

Кінець таблиці 4.1 – Опис основних функцій ПЗ

| № п.п | 1) Назва функції | Опис функції |
|---|---|---|
| 23 | tableWidget() | Функція формує таблицю правил |
| 24 | setItem() | Функція записує елементи таблиці правил фаєрволу |
| 26 | about(this) | Функція видає повідомлення про результат операції |
| 27 | qt_static_metacall() | Функція обробки статичних об’єктів графічного інтерфейсу |
| 28 | staticMetaObjectExtraData() | Функція створення та оперування об’єктами графічного інтерфейсу |
| 29 | getStaticMetaObject() | Функція виклику статичних об’єктів графічного інтерфейсу |
| 30 | qt_metacast() | Функція передачі даних між об’єктами графічного інтерфейсу |
| 31 | qt_meta_stringdata_MainWindow() | Функція передачі та обробки даних головного вікна графічного інтерфейсу |
| 32 | InvokeMetaMethod() | Функція роботи основних методів Qt-бібліотеки |

# ВИСНОВКИ

Оптимізація колонії мурашок була і залишається плідною парадигмою для розробки ефективних алгоритмів рішення комбінаторної оптимізації. Після досліджень було продемонстровано як ефективність його застосування, так і теоретичні обґрунтування, що робить ОПМ однією з найуспішніших парадигм в метаевристичні області.

За результатами проведеного дослідження запропоновано алгоритм оптимізації корпоративної комп'ютерної мережі за допомогою мурашиних алгоритмів. Запропонований метод показав себе краще ніж стандартний алгоритм оптимізації АСО.

У першому розділі було розглянуто Основні структурно складові корпоративної мережі, необхідне обладнання для побудови ККМ, переваги, можливості, складнощі при створенні корпоративних мереж.

У другому розділі розглянуто саму реалізації корпоративної мережі, вибір і аналіз обладнання, створення концептуальної схеми мережі.

У третьому розділі проведено мільти фракційний аналіз описані шляхи оптимізації за допомогою колоній мурашок, описані особливості і системи КМ. Описані часті проблеми оптимізації АСО а також механізми оновлення маршруту.

У четвертому розділі запропоновано підхід для балансування навантаження в розподілених системах на основі численних колоній мурашок була запропонована оптимізація. Використання кількох гнізд, або колоній мурашок у процесі пошуку, допомогло підвищити швидкість обміну інформацією по всіх вузлах системи. Крім того, динамічний обмін інформацією та її повне розповсюдження - це інші основні характеристики, що відрізняють цей підхід. Результати показали ефективність запропонованої моделі в порівнянні зі стандартним алгоритмам викрадення роботи з точки зору кількості зайнятих вузлів та витраченого часу для досягнення ефективності 50%

За темою дипломної роботи опубліковані матеріали у XII ВСЕУКРАЇНСЬКІЙ НАУКОВО-ПРАКТИЧНІЙ КОНФЕРЕНЦІЇ АПКН 2020

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. M. Dorigo, G. D. Caro, and L. M. Theraulaz, Inspiration for optimization from social insect behavior. *Nature*, 2000. vol. 406. pp. 39-42.

a. Sydney, C. Scoglio, P. Schumm and R.E. Kooij, Elasticity: Topological characterization of robustness in complex networks, *IEEE/ACM Bionetics*, 2008. vol. 406. pp. 39-42.

2. Y. Singer, *Dynamic measure of network robustness*, 2006. vol. 406. pp. 39-42.

3. Alexander Veremygy and Vladimir Boginski, *Robustness and Strong Attack Tolerence of Low Diameter Networks*, January 2012. vol. 406. pp. 39-42.

4. W. John, *Chinneck Practical Optimization: a Gentle Introduction*. vol. 406. pp. 39-42.

5. V.L. Patil, *Computer Network Optimization technical report Trinity College of Engineering and Research*, 2012. vol. 406. pp. 39-42.

6. R. Jovanovic and M. Tuba, *Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem", Computer Science and Information Systems*, 2013. vol. 10, no. 1, pp. 133-149,

7. S. Gajjar, M. Sarkar and K. Dasgupta, *"Famacrow: Fuzzy and ant colony optimization based combined mac routing and unequal clustering cross-layer protocol for wireless sensor networks"*, Applied Soft Computing, , 2016. vol. 43, pp. 235-247

8. K. JIANG, M. LI and H. ZHANG, *Improved ant colony algorithm for travelling salesman problem, Journal of Computer Applications*, 2015. , vol. S2, pp.

9. Q. Cai, D. Zhang, W. Zheng and S. C. Leung, *A new fuzzy time series forecasting model combined with ant colony optimization and auto-regression*, Knowledge-Based Systems, 2015vol. 74, pp. 61-68,.

10. N. Jalil, S. Julai and R. Ramli, *Parametric modelling of flexible plate structures using continuous ant colony optimization, Journal of Simulation*, vol. 9, no. 3, , 2015 pp. 223-231.

11. W. Zheng, S. Mo, Y. Qu, X. Jin, J. Zhou, P. Duan, et al., *Pattern learning based parallel ant colony optimization* 2017, pp. 497-502.

a. Rashno, B. Nazari, S. Sadri and M. Saraee, *Effective pixel classification of mars images based on ant colony optimization feature selection and extreme learning machine Neurocomputing*, 2017.vol. 226, pp. 66-79,

12. R. I. Chang, S. Y. Lin and Y. Hung, *Particle swarm optimization with query-based learning for multi-objective power contract problem*, Expert Systems with Applications2012., vol. 39, no. 3, pp. 3116-3126,

a. K. Z. Alsaedi, R. Ghazali and M. M. Deris, An efficient multi join query optimization for relational *database management system using two phase artificial bess colony algorithm*, 2015.

13. M. Alamery, A. Faraahi, H. H. S. Javadi, S. Nourossana and H. Erfani, "*Multi-join query optimization using the bees algorithm*", Distributed Computing and Artificial Intelligence - International Symposium Dcai 2010, pp. 449-457, 7–10 September

14. Hao Yuan, Changbing Li and Maokang Du, *Resource Scheduling of Cloud Computing for Node* of Wireless Sensor Network Based on Ant Colony Algorithm", Information Technology Journal2012., vol. 28, pp. 1638-1643, August

15. Anthony H. Dekker and Bemard D. Colbert, "*Network Robustness and Graph Topology*", Research and Practice in Information Technology, vol. 26.

16. Priscila V.S.Z. Capriles, Leonardo Goliatt da Fonseca and Helio, J. C. Barbosa, *Ant Colony Algorithms Applied To Discrete Optimization Problems*.

17. Hadi Rezazad, "Computer Network Optimization" in , Willey Interdisplinary Reviews: Computational Statastics, vol. 3, pp. 34-46.

18. M. Dorigo, V. Maniezzo, and A. Colomi, "*Ant system: optimization by a colony of cooperating agents*", IEEE trans. on System, Man, and Cybernetics-Part B: Cybernetics1993., vol. 26, no. 1, pp. 29-41, Feb.

19. D.Costa and A. Hertz, *Ants can color graph,*J. Oper. Res. Soc1997.., vol. 48, no. 3, pp. 295-305, Mar.

20. T. Ma, Q. Yan, W. Liu, D. Guan, and S. Lee, *Grid task scheduling: algorithm review*, *IETE Technical Review*2011., vol. 28, no. 2, pp. 158–167,

21. W. T. Zaumen, S. Vutukury, and J.J. Garcia-Luna-Aceves, *Load-balanced anycast routing in computer networks,* in proceedings of Fifth IEEE Symposium on Computers and Communications, 2000pp. 566-574, 3-6 July

22. R. S. Parpinelli, H. S. Lopes, A. A. Freitas, *Data mining with an ant colony optimization algorithm,*" IEEE Trans. Evolutionary Computation 2002., vol. 6, no. 4, pp. 321-332,Aug.

23. K. M. Sim and W. H. Sun, *Ant colony optimization for routing and load-balancing: survey and new directions,* IEEE trans. on System, Man, and Cybernetics-Part A: Systems and Humans2003., vol. 33, No. 5, Sept.

24. Arianyan, E., Taheri, H., & Sharifian, S. (2015). *Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers. Computers and Electrical Engineering*2015.05.006, 47, 222–240. doi: 10.1016/j.compeleceng.

25. M. Dorigo, T. S. Zel, *Ant colony optimization*. Bradford, 2004.

26. M. Dorigo, G. D. Caro, The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, F. Glover (Eds.), *New Ideals in Optimization, McGraw-Hill, 1999.. S. Ahn et al. Evaluation of TCP Vegas: Emulation and Experiment*. In Proc. of SIGCOMM 1995.'95, pages 185--195, Aug.

a. Beloglazov and R. Buyya, *Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers*, Concurrency and Computation: Practice and Experience, 2012.vol. 24, no. 13, pp. 1397–1420,

27. M. Escheikh, K. Barkaoui, and H. Jouini, *Versatile workload-aware power management performability analysis of server virtualized systems*, The Journal of Systems and Software2017., vol. 125, pp. 365–379,

28. Y. Xia, M. Zhou, X. Luo, S. Pang, and Q. Zhu, *A stochastic approach to analysis of energy-aware DVS-enabled cloud datacenters,* IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2015.vol. 45, no. 1, pp. 73–83,

29. D. Rossi, V. Tenentes, S. Khursheed, and B. M. Al-Hashimi, *BTI and leakage aware dynamic voltage scaling for reliable low power cache memories*, in Proceedings of the 21st IEEE International On-Line Testing Symposium, IOLTS 2015, pp. 194–199, July

30. X. Chen, Z. Xu, H. Kim et al., "*Dynamic voltage and frequency scaling for shared resources in multicore processor designs*," in Proceedings of the 50th Annual Design Automation Conference, DAC 2013, p. 114, June

31. Y. Singer, *Dynamic measure of network robustness*, 2006. vol. 406. pp. 39-42.

32. Alexander Veremygy and Vladimir Boginski, *Robustness and Strong Attack Tolerence of Low Diameter Networks*, January 2012. vol. 406. pp. 39-42.

33. W. John, *Chinneck Practical Optimization: a Gentle Introduction*. vol. 406. pp. 39-42.

34. V.L. Patil, *Computer Network Optimization technical report Trinity College of Engineering and Research*, 2012. vol. 406. pp. 39-42.

35. R. Jovanovic and M. Tuba, *Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem", Computer Science and Information Systems*, 2013. vol. 10, no. 1, pp. 133-149,

36. S. Gajjar, M. Sarkar and K. Dasgupta, *"Famacrow: Fuzzy and ant colony optimization based combined mac routing and unequal clustering cross-layer protocol for wireless sensor networks"*, Applied Soft Computing, , 2016. vol. 43, pp. 235-247

37. K. JIANG, M. LI and H. ZHANG, *Improved ant colony algorithm for travelling salesman problem, Journal of Computer Applications*, 2015. , vol. S2, pp.

38. Q. Cai, D. Zhang, W. Zheng and S. C. Leung, *A new fuzzy time series forecasting model combined with ant colony optimization and auto-regression*, Knowledge-Based Systems, 2015vol. 74, pp. 61-68,.

39.

40. M. Dorigo, V. Maniezzo, and A. Colomi, "*Ant system: optimization by a colony of cooperating agents*", IEEE trans. on System, Man, and Cybernetics-Part B: Cybernetics1993., vol. 26, no. 1, pp. 29-41, Feb.

41. D.Costa and A. Hertz, *Ants can color graph,*J. Oper. Res. Soc1997.., vol. 48, no. 3, pp. 295-305, Mar.

42. T. Ma, Q. Yan, W. Liu, D. Guan, and S. Lee, *Grid task scheduling: algorithm review, IETE Technical Review*2011., vol. 28, no. 2, pp. 158–167,

43. W. T. Zaumen, S. Vutukury, and J.J. Garcia-Luna-Aceves, *Load-balanced anycast routing in computer networks,* in proceedings of Fifth IEEE Symposium on Computers and Communications, 2000pp. 566-574, 3-6 July

44. R. S. Parpinelli, H. S. Lopes, A. A. Freitas, *Data mining with an ant colony optimization algorithm,*" IEEE Trans. Evolutionary Computation 2002., vol. 6, no. 4, pp. 321-332,Aug.

45. K. M. Sim and W. H. Sun, *Ant colony optimization for routing and load-balancing: survey and new directions*, IEEE trans. on System, Man, and Cybernetics-Part A: Systems and Humans2003., vol. 33, No. 5, Sept.

46. Arianyan, E., Taheri, H., & Sharifian, S. (2015). *Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers. Computers and Electrical Engineering*2015.05.006, 47, 222–240. doi: 10.1016/j.compeleceng.

47. M. Dorigo, T. S. Zel, *Ant colony optimization*. Bradford, 2004.

48. M. Dorigo, G. D. Caro, The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, F. Glover (Eds.), *New Ideals in Optimization, McGraw-Hill, 1999.. S. Ahn et al. Evaluation of TCP Vegas: Emulation and Experiment*. In Proc. of SIGCOMM 1995.'95, pages 185--195, Aug.

a. Beloglazov and R. Buyya, *Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers*, Concurrency and Computation: Practice and Experience, 2012.vol. 24, no. 13, pp. 1397–1420,

49. M. Escheikh, K. Barkaoui, and H. Jouini, *Versatile workload-aware power management performability analysis of server virtualized systems*, The Journal of Systems and Software2017., vol. 125, pp. 365–379,

50. Y. Xia, M. Zhou, X. Luo, S. Pang, and Q. Zhu, *A stochastic approach to analysis of energy-aware DVS-enabled cloud datacenters*, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2015.vol. 45, no. 1, pp. 73–83,

51. D. Rossi, V. Tenentes, S. Khursheed, and B. M. Al-Hashimi, *BTI and leakage aware dynamic voltage scaling for reliable low power cache memories*, in Proceedings of the 21st IEEE International On-Line Testing Symposium, IOLTS 2015, pp. 194–199, July

52. X. Chen, Z. Xu, H. Kim et al., "*Dynamic voltage and frequency scaling for shared resources in multicore processor designs*," in Proceedings of the 50th Annual Design Automation Conference, DAC 2013, p. 114, June

53. Y. Singer, *Dynamic measure of network robustness*, 2006. vol. 406. pp. 39-42.

a. Sydney, C. Scoglio, P. Schumm and R.E. Kooij, Elasticity: Topological characterization of robustness in complex networks, *IEEE/ACM Bionetics*, 2008. vol. 406. pp. 39-42.

54. Y. Singer, *Dynamic measure of network robustness*, 2006. vol. 406. pp. 39-42.

55. Alexander Veremygy and Vladimir Boginski, *Robustness and Strong Attack Tolerence of Low Diameter Networks*, January 2012. vol. 406. pp. 39-42.

56. W. John, *Chinneck Practical Optimization: a Gentle Introduction*. vol. 406. pp. 39-42.

57. V.L. Patil, *Computer Network Optimization technical report Trinity College of Engineering and Research*, 2012. vol. 406. pp. 39-42.

58. R. Jovanovic and M. Tuba, *Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem", Computer Science and Information Systems*, 2013. vol. 10, no. 1, pp. 133-149,

59. S. Gajjar, M. Sarkar and K. Dasgupta, *"Famacrow: Fuzzy and ant colony optimization based combined mac routing and unequal clustering cross-layer protocol for wireless sensor networks"*, Applied Soft Computing, , 2016. vol. 43, pp. 235-247

60. K. JIANG, M. LI and H. ZHANG, *Improved ant colony algorithm for travelling salesman problem, Journal of Computer Applications*, 2015. , vol. S2, pp.

61. Q. Cai, D. Zhang, W. Zheng and S. C. Leung, *A new fuzzy time series forecasting model combined with ant colony optimization and auto-regression*, Knowledge-Based Systems, 2015vol. 74, pp. 61-68,.

# ДОДАТОК А

## ЛІСТИНГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОПТИМІЗАЦІЇ КОРПОРАТИВНГІ МЕРЕЖІ МУРАШИНИМ АЛГОРИТМОМ

Модуль «Оптимізація мурашиним алгоритмом».

```
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

namespace Experimenter
{
   internal class Ant_Colony_Optim
   {
       private Random random = new Random(0);

       public  double alpha{get; set;}

       public double beta{get; set;}

       public  double rho{ get; set; }
       public double Q{ get; set; }

       public  int numCities{ get; set; }

       public int numAnts{ get; set; }
       public int  maxTime{ get; set; }

       public int  maxRand;



       public double BestInitial {get; set;}
       public double Best { get; set; }
       public int Time {get; set;}
       public int[] Trail { get; set; }


       public Ant_Colony_Optim(string Dir, double Alpha, double
Beta, double Rho, double Q_ , int NumCities, int NumAnts, int
MaxTime)
```

```
        {
            alpha = Alpha;
            beta = Beta;
            rho = Rho;
            Q = Q_ ;
            numCities = NumCities;
            numAnts = NumAnts;
            maxTime = MaxTime;


        try
            {

                double[][] dists = ReadDir(Dir);

                int[][]  ants = InitAnts(numAnts, numCities); //
initialize ants to random trails

                int[]  bestTrail = BestTrail(ants, dists); //
determine the best initial trail



                double bestLength = Length(bestTrail, dists); //
the length of the best initial trail

                BestInitial = bestLength;

                double[][]  pheromones = InitPheromones(numCities);

                int time = 0;

                while (time < maxTime)
                {
                    UpdateAnts(ants, pheromones, dists);
                    UpdatePheromones(pheromones, ants, dists);

                    int[]  currBestTrail = BestTrail(ants, dists);

                    double currBestLength = Length(currBestTrail,
dists);
                    if (currBestLength < bestLength)
                    {
                        bestLength = currBestLength;
                        bestTrail = currBestTrail;
                        Time = time;
                    }
```

```
                ++time;
            }

            Trail = bestTrail; Best = bestLength;

        }
        catch (Exception ex)
        {
            if (MessageBox.Show(ex.Message,"Runtime error",
MessageBoxButton.OK) == MessageBoxResult.OK)
            {
                Application.Current.Shutdown();
            }
        }
    }

    double[][] ReadDir(string p)
    {
        string[] filePaths = Directory.GetFiles(p);


        double[][] dists = new double[numCities][];
        for (int i = 0; i < dists.Length; ++i)
            dists[i] = new double[numCities];

        foreach (string s in filePaths)
        {
            string json = File.ReadAllText(s);
            ParseJson parse = new ParseJson(json);

            int index = s.LastIndexOf("\\");
            int index_ = s.LastIndexOf("_");
            int index__ = s.LastIndexOf(".");

            int i = Convert.ToInt32( s.Substring(index + 1,
index_ - index - 1));
            int j = Convert.ToInt32(s.Substring(index_ + 1 ,
index__ - index_ - 1));

            dists[i][j] = parse.Distance;
        }

        return dists;
    }

     int[][] InitAnts(int numAnts, int numCities)
    {
        int[][] ants = new int[numAnts][];
```

```
            for (int k = 0; k < numAnts; ++k)
            {
                int start = random.Next(0, numCities);
                ants[k] = RandomTrail(start, numCities);
            }
            return ants;
        }

         int[] RandomTrail(int start, int numCities) // helper for
InitAnts
        {
            int[] trail = new int[numCities];

            for (int i = 0; i < numCities; ++i) { trail[i] = i; }
// sequential

            for (int i = 0; i < numCities; ++i) // Fisher-Yates
shuffle
            {
                int r = random.Next(i, numCities);
                int tmp = trail[r]; trail[r] = trail[i]; trail[i] =
tmp;
            }

            int idx = IndexOfTarget(trail, start); // put start at
[0]
            int temp = trail[0];
            trail[0] = trail[idx];
            trail[idx] = temp;

            return trail;
        }

         int IndexOfTarget(int[] trail, int target) // helper for
RandomTrail
        {
            for (int i = 0; i < trail.Length; ++i)
            {
                if (trail[i] == target)
                    return i;
            }
            throw new Exception("Target not found in
IndexOfTarget");
        }

         double Length(int[] trail, double[][] dists) // total
length of a trail
        {
            double result = 0.0;
```

```
            for (int i = 0; i < trail.Length - 1; ++i)
                result += Distance(trail[i], trail[i + 1], dists);
            return result;
        }

        // ------------------------------------------------------
-----------------------------------

         int[] BestTrail(int[][] ants, double[][] dists) // best
trail has shortest total length
        {
            double bestLength = Length(ants[0], dists);
            int idxBestLength = 0;
            for (int k = 1; k < ants.Length; ++k)
            {
                double len = Length(ants[k], dists);
                if (len < bestLength)
                {
                    bestLength = len;
                    idxBestLength = k;
                }
            }
            int numCities = ants[0].Length;
            int[] bestTrail = new int[numCities];
            ants[idxBestLength].CopyTo(bestTrail, 0);
            return bestTrail;
        }

        // ------------------------------------------------------
-----------------------------------

        double[][] InitPheromones(int numCities)
        {
            double[][] pheromones = new double[numCities][];
            for (int i = 0; i < numCities; ++i)
                pheromones[i] = new double[numCities];
            for (int i = 0; i < pheromones.Length; ++i)
                for (int j = 0; j < pheromones[i].Length; ++j)
                    pheromones[i][j] = 0.00000000000001; //
otherwise first call to UpdateAnts -> BuiuldTrail -> NextNode ->
MoveProbs => all 0.0 => throws
            return pheromones;
        }

        // ------------------------------------------------------
-----------------------------------

        void UpdateAnts(int[][] ants, double[][] pheromones,
double[][] dists)
```

```
        {
            int numCities = pheromones.Length;
            for (int k = 0; k < ants.Length; ++k)
            {
                int start = random.Next(0, numCities);
                int[] newTrail = BuildTrail(k, start, pheromones,
dists);
                ants[k] = newTrail;
            }
        }

        int[] BuildTrail(int k, int start, double[][] pheromones,
double[][] dists)
        {
            int numCities = pheromones.Length;
            int[] trail = new int[numCities];
            bool[] visited = new bool[numCities];
            trail[0] = start;
            visited[start] = true;
            for (int i = 0; i < numCities - 1; ++i)
            {
                int cityX = trail[i];
                int next = NextCity(k, cityX, visited, pheromones,
dists);
                trail[i + 1] = next;
                visited[next] = true;
            }
            return trail;
        }

        int NextCity(int k, int cityX, bool[] visited, double[][]
pheromones, double[][] dists)
        {
            // for ant k (with visited[]), at nodeX, what is next
node in trail?
            double[] probs = MoveProbs(k, cityX, visited,
pheromones, dists);

            double[] cumul = new double[probs.Length + 1];
            for (int i = 0; i < probs.Length; ++i)
                cumul[i + 1] = cumul[i] + probs[i]; // consider
setting cumul[cuml.Length-1] to 1.00

            double p = random.NextDouble();

            for (int i = 0; i < cumul.Length - 1; ++i)
                if (p >= cumul[i] && p < cumul[i + 1])
                    return i;
```

```
            throw new Exception("Failure to return valid city in
NextCity");
        }

        double[] MoveProbs(int k, int cityX, bool[] visited,
double[][] pheromones, double[][] dists)
        {
            // for ant k, located at nodeX, with visited[], return
the prob of moving to each city
            int numCities = pheromones.Length;
            double[] taueta = new double[numCities]; // inclues
cityX and visited cities
            double sum = 0.0; // sum of all tauetas
            for (int i = 0; i < taueta.Length; ++i) // i is the
adjacent city
            {
                if (i == cityX)
                    taueta[i] = 0.0; // prob of moving to self is 0
                else if (visited[i] == true)
                    taueta[i] = 0.0; // prob of moving to a visited
city is 0
                else
                {
                    taueta[i] = Math.Pow(pheromones[cityX][i],
alpha) * Math.Pow((1.0 / Distance(cityX, i, dists)), beta); //
could be huge when pheromone[][] is big
                    if (taueta[i] < 0.00000000000001)
                        taueta[i] = 0.00000000000001;
                    else if (taueta[i] > (double.MaxValue /
(numCities * 100)))
                        taueta[i] = double.MaxValue / (numCities *
100);
                }
                sum += taueta[i];
            }

            double[] probs = new double[numCities];
            for (int i = 0; i < probs.Length; ++i)
                probs[i] = taueta[i] / sum; // big trouble if sum =
0.0
            return probs;
        }

        // ---------------------------------------------------------
-----------------------------------

        void UpdatePheromones(double[][] pheromones, int[][] ants,
double[][] dists)
        {
            for (int i = 0; i < pheromones.Length; ++i)
```

```
            {
                for (int j = i + 1; j < pheromones[i].Length; ++j)
                {
                    for (int k = 0; k < ants.Length; ++k)
                    {
                        double length = Length(ants[k], dists); //
length of ant k trail
                        double decrease = (1.0 - rho) *
pheromones[i][j];
                        double increase = 0.0;
                        if (EdgeInTrail(i, j, ants[k]) == true)
increase = (Q / length);

                        pheromones[i][j] = decrease + increase;

                        if (pheromones[i][j] < 0.0000001)
                            pheromones[i][j] = 0.0000001;
                        else if (pheromones[i][j] > 100000.0)
                            pheromones[i][j] = 100000.0;

                        pheromones[j][i] = pheromones[i][j];
                    }
                }
            }
        }

         bool EdgeInTrail(int cityX, int cityY, int[] trail)
        {
            // are cityX and cityY adjacent to each other in
trail[]?
            int lastIndex = trail.Length - 1;
            int idx = IndexOfTarget(trail, cityX);

            if (idx == 0 && trail[1] == cityY) return true;
            else if (idx == 0 && trail[lastIndex] == cityY) return
true;
            else if (idx == 0) return false;
            else if (idx == lastIndex && trail[lastIndex - 1] ==
cityY) return true;
            else if (idx == lastIndex && trail[0] == cityY) return
true;
            else if (idx == lastIndex) return false;
            else if (trail[idx - 1] == cityY) return true;
            else if (trail[idx + 1] == cityY) return true;
            else return false;
        }

        // ------------------------------------------------------
-----------------------------------
```

```
         double Distance(int cityX, int cityY, double[][] dists)
        {
            return dists[cityX][cityY];
        }

        // ------------------------------------------------------------
-------------------------------------


    }
}
```

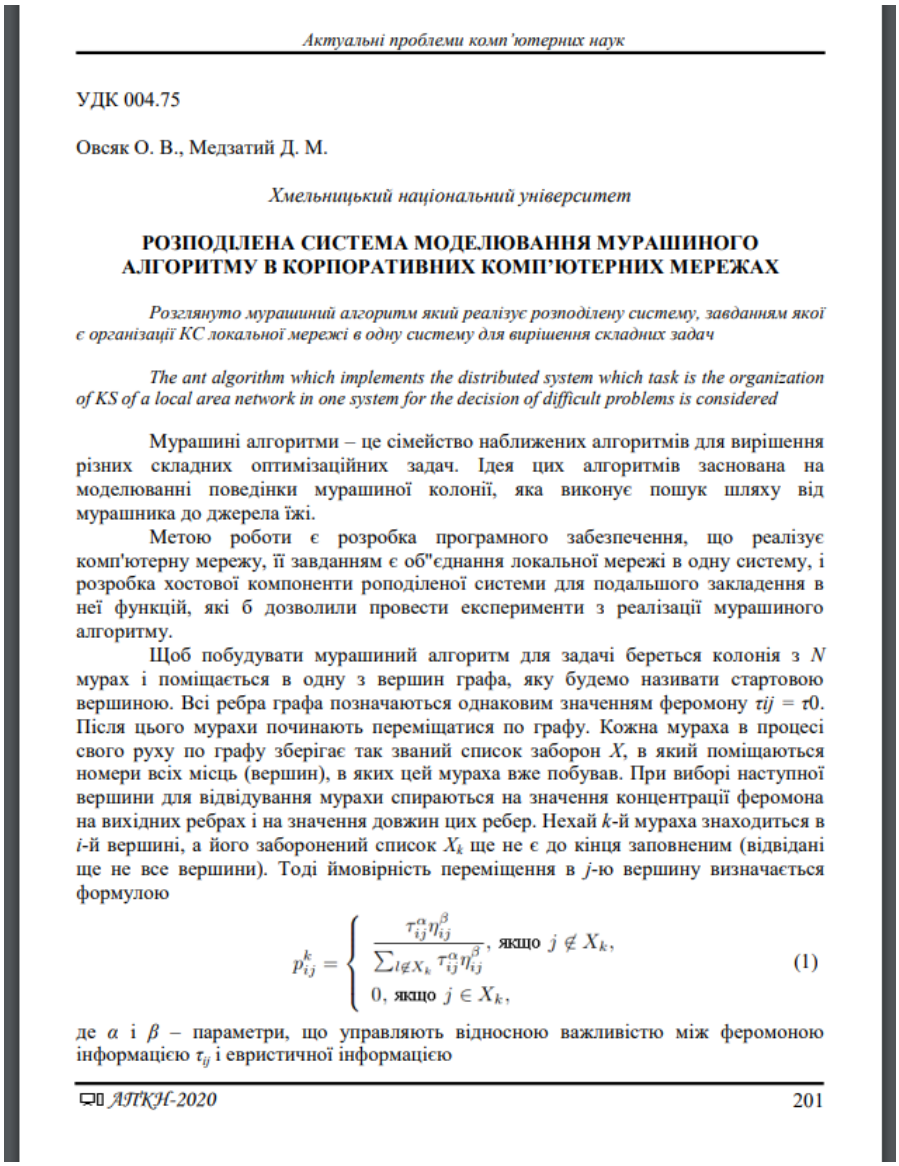
# ДОДАТОК Б

(обов’язковий)

## СТАТТІ ЗА РЕЗУЛЬТАТАМИ ДОСЛІДЖЕННЯ

Овсяк О.В., Медзатий Д.М. Розподілена система моделювання мурашиного алгоритму в комп’ютерних корпоративних мережах

Збірник наукових праць за матеріалами XII Всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2020 С. 201-202.





УДК 004.75

Овсяк О. В., Медзатий Д. М.

*Хмельницький національний університет*

**РОЗПОДІЛЕНА СИСТЕМА МОДЕЛЮВАННЯ МУРАШИНОГО АЛГОРИТМУ В КОРПОРАТИВНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ**

*Розглянуто мурашиний алгоритм який реалізує розподілену систему, завданням якої є організації КС локальної мережі в одну систему для вирішення складних задач*

*The ant algorithm which implements the distributed system which task is the organization of KS of a local area network in one system for the decision of difficult problems is considered*

Мурашині алгоритми – це сімейство наближених алгоритмів для вирішення різних складних оптимізаційних задач. Ідея цих алгоритмів заснована на моделюванні поведінки мурашиної колонії, яка виконує пошук шляху від мурашника до джерела їжі.

Метою роботи є розробка програмного забезпечення, що реалізує комп'ютерну мережу, її завданням є об"єднання локальної мережі в одну систему, і розробка хостової компоненти роподіленої системи для подальшого закладення в неї функцій, які б дозволили провести експерименти з реалізації мурашиного алгоритму.

Щоб побудувати мурашиний алгоритм для задачі береться колонія з $N$ мурах і поміщається в одну з вершин графа, яку будемо називати стартовою вершиною. Всі ребра графа позначаються однаковим значенням феромону $\tau ij = \tau 0$. Після цього мурахи починають переміщатися по графу. Кожна мураха в процесі свого руху по графу зберігає так званий список заборон $X$, в який поміщаються номери всіх місць (вершин), в яких цей мураха вже побував. При виборі наступної вершини для відвідування мурахи спираються на значення концентрації феромона на вихідних ребрах і на значення довжин цих ребер. Нехай $k$-й мураха знаходиться в $i$-й вершині, а його заборонений список $X_k$ ще не є до кінця заповненим (відвідані ще не все вершини). Тоді ймовірність переміщення в $j$-ю вершину визначається формулою

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l \notin X_k} \tau_{ij}^{\alpha}\eta_{ij}^{\beta}}, \text{ якщо } j \notin X_k, \\ 0, \text{ якщо } j \in X_k, \end{cases} \quad (1)$$

де $\alpha$ і $\beta$ – параметри, що управляють відносною важливістю між феромоною інформацією $\tau_{ij}$ і евристичної інформацією

$$\eta_{ij} = \frac{1}{d_{ij}}. \quad (2)$$

Якщо ж у $k$-го мурахи не залишилося не відвіданих вершин, то він, вже не використовуючи імовірнісного вибору, просто повертається в стартову вершину.

Після того, як всі мурахи повернуться додому, зробивши рівно по $n$ переміщень, проводиться оновлення значень концентрації феромона на ребрах графа. Кожна мураха оновлює тільки ті ребра, по яким він переміщався. Концентрація феромону на ребрі $(i, j)$ перераховується за формулою:

$$\tau_{ij} \leftarrow \rho\tau_{ij} + \sum_{k=1}^{N} \Delta\tau_{ij}^{k}, \quad (3)$$

де $\rho$ – коефіцієнт випаровування феромону, $\Delta\tau_{ij}^{k}$ – кількість феромону, що залишається на цьому ребрі $k$-им мурахою:

$$\Delta T \frac{k}{ij} = \begin{cases} \text{Q/Dk, якщо к-й мураха проходив по ребру (i, j),} \\ \\ \text{0, інакше} \end{cases}$$

де $Q$ – деяка фіксована величина.

Таким чином, чим коротше шлях $k$-го мурашки, тим більше феромона він залишить на ребрах, по яким він проходив.

Описані дії (переміщення по графу і оновлень феромонів) представляють собою одну ітерацію мурашиного алгоритму. Ітерації повторюються до тих пір, поки не виявиться виконаним котрійсь із критеріїв зупинки алгоритму - вичерпано число ітерацій, досягне встановленої точність, отриманий єдиний шлях (алгоритм зійшовся до деякого рішенням).

Розглянутий мурашиний алгоритм може бути реалізований з використанням розподіленої системи, завданням якої є організації КС локальної мережі в одну систему.

**Перелік посилань**


1. https://www.sciencedirect.com/science/article/pii/S0895717710002116
2. https://msn.khnu.km.ua/pluginfile.php/333045/mod_resource/content/2/Лабораторна%20робота№3.pdf
3. https://uk.wikipedia.org/wiki/Мурашиний_алгоритм
4. https://wiki.loginom.ru/articles/ant-colony-optimization.html

# ДОДАТОК В

# ПРЕЗЕНТАЦІЯ ДОПОВІДІ

## Розподілена система моделювання мурашиного алгоритму в корпоративних комп'ютерних мережах

Студент групи КІ2м-19-1
Овсяк О.В.

Науковий керівник
к.т.н., доцент Медзатий Д.М.

Хмельницький 2021

## Об'єкт, предмет та мета дослідження

**Метою роботи** є оптимізація шляхом досягнення результатів балансування навантаження, іншими словами, мінімізація затримки та часу відгуку при одночасному збільшенні пропускної здатності простими словами поліпшити продуктивність PC.

**Об'єктом дослідження** є процес оптимізації навантаження розподілених систем.

**Предметом дослідження** є розподілена система моделювання мурашиного алгоритму в корпоративних комп'ютерних мережах.

# Наукова новизна

**Удосконалено метод** оптимізації мережі мурашиним алгоритмом, шляхом балансування навантаження, тобто мінімізування затримки та часу відгуку при одночасному збільшенні пропускної здатності.

У цьому експерименті запропоновано підхід для балансування навантаження в розподілених системах на основі численних колоній мурашок була запропонована оптимізація. Використання кількох гнізд, або колоній мурашок у процесі пошуку, допомогло підвищити швидкість обміну інформацією по всіх вузлах системи. Крім того, динамічний обмін інформацією та її повне розповсюдження - це інші основні характеристики, що відрізняють цей підхід. Результати показали ефективність запропонованої моделі в порівнянні зі стандартним алгоритмам викрадення роботи з точки зору кількості зайнятих вузлів та витраченого часу для досягнення ефективності на 15%

# Практичне значення одержаних результатів

**Практична цінність** отриманих результатів. В результаті виконання наукового дослідження розроблений алгоритм оптимізації навантаження на розподілену систему що в подальшому дає широкий спектр можливостей з оптимізацією мереж для покращення ефективності.

У результаті виконуваного дослідження запропоновано підхід для балансування навантаження в розподілених системах на основі численних колоній мурашок була запропонована оптимізація. Використання кількох гнізд, або колоній мурашок у процесі пошуку, допомогло підвищити швидкість обміну інформацією по всіх вузлах системи. Крім того, динамічний обмін інформацією та її повне розповсюдження - це інші основні характеристики, що відрізняють цей підхід. Результати показали ефективність запропонованої моделі в порівнянні зі стандартним алгоритмам викрадення роботи з точки зору кількості зайнятих вузлів та витраченого часу для досягнення ефективності 50%

# Постановка задачі дослідження

- Дослідити особливості оптимізації комп`ютерних корпоративних мереж;
- Створити модель розподіленої системи;
- Аналіз існуючих методів оптимізації мурашиних алгоритмів;
- Провести експеримент із запропонованою системою оптимізації мережі;

# Ефективність розподілених систем

Існує дві основні концепції, які обмежують ефективність розподіленої системи:

Затримка : час, необхідний для передачі повідомлення з одного місця в інше в розподіленій системі.

Пропускна здатність : кількість даних, яка може бути передана за одиницю часу в стабільному стані.

Крім цього, тут слід включити два провідні показники ефективності:

Час відповіді : час, необхідний для отримання результату після подання завдання на обробку системою.

Пропускна здатність : здатність системи витримувати великі навантаження. Іншими словами, кількість виконаних завдань за одиницю часу.

Метою розробленої розподіленої обчислювальної системи є досягнення результатів балансування навантаження, іншими словами, мінімізація затримки та часу відгуку при одночасному збільшенні пропускної здатності простими словами поліпшити продуктивність PC . Для цього необхідно здійснити певні оптимізації системи.

# Обраний метод оптимізації

**Запропоновано метод балансування навантаження** - щоб отримати максимальну продуктивність, кожному апарату в системі потрібно призначити однаковий обсяг роботи. Оскільки різні завдання не обов'язково вимагають однакової кількості обчислень, роботи чи часу, балансування навантаження не означає просто прохання кожної машини виконати іменну кількість завдань. Натомість існує два різні підходи до збалансування навантаження обчислювальної системи. Якщо різні розміри завдань відомі заздалегідь, навантаження можна статично збалансувати під час компіляції; якщо, однак, різний розмір завдання невідомий, завдання призначаються різним процесорам динамічно під час виконання.

Паралельно всі машини, які виконують різні частини будь-яких обчислень, повинні для оптимальної продуктивності виконувати різні обчислення одночасно. Це просто оптимізація, заснована на здоровому глузді, якщо є два процесори, які працюють разом, щоб виконати якийсь великий розрахунок, який потрібно об'єднати, а потім ще раз обробити, третій аналіз вимагає завершення перших двох обчислень перед початком. Таким чином, якщо дві обчислювальні машини працюють одночасно, час, витрачений на очікування закінчення обчислення одним процесором, тоді як інший сидить склавши руки, чекаючи початку нового завдання, буде мінімізований.

При використанні розподіленої обчислювальної системи, де всі обчислення просто виконуються, коли користувач не використовує машину, неможливо мати повністю одночасну систему. Це просто питання практичності, немає можливості отримати оптимальний стан, коли всі обчислення повністю синхронізовані між усіма комп'ютерами в розподіленій системі.

Нарешті, у розподіленій обчислювальній системі завжди будуть накладні витрати на систему. Різні машини повинні розмовляти між собою, і в результаті накладні витрати стають функцією затримки та пропускної здатності системи. Крім того, різним процесорам, можливо, доведеться повторити деякі обчислення, виконані іншими машинами, локально, просто для виконання своїх непомітних завдань. Якщо ці різні завдання виконувались на одній машині послідовно, ці повторювані обчислення могли не знадобитися.

# Постановка експерименту

Оптимізація колонії мурашок (ACO) забезпечує інструмент мета евристичної оптимізації та модель колективного інтелекту для кількох додатків, таких як маршрутизація та балансування навантаження. У літературі знайдено багато робіт з використання ACO в балансуванні навантаження. Однак, наскільки мені відомо, не було роботи щодо балансування навантаження в розподілених системах з ACO.

У даній магістерській роботі був представлений експеримент з ACO для балансування навантаження в розподілених системах. Цей експеримент повністю розподілений, в якому інформація динамічно оновлюється при кожному русі мурахи. Буде прийнята парадигма кількох колоній, така що кожен вузол буде посилати кольорову колонію по всій мережі. У цьому дослідженні кольорові колонії мурашок використовуються для запобігання руху мурах одного гнізда за тим самим маршрутом і, отже, примусового їх розподілу по всіх вузлах системи, і кожен мураха діє як мобільний агент, який несе нещодавно оновлену інформацію про балансування навантаження до наступного відвіданого вузла. Висновок: Нарешті, ефективність запропонованого алгоритму ACO порівнюється з підходом викрадення роботи для балансування навантаження в розподілених системах.

# Аналіз розподіленої системи балансування

Розподілена система балансування навантаження все ще залишається активною областю досліджень, в якій шляхом балансування навантаження намагається поліпшити продуктивність розподіленої системи за допомогою обчислювальної потужності всієї системи для згладжування періодів високих перевантажень в окремих вузлах , це робиться шляхом перенесення частини робочого навантаження сильно завантажених вузлів на інші вузли для обробки. Рішення щодо того, як збалансувати навантаження між вузлами, є або статичними або динамічний. Статичне рішення не залежить від поточного стану системи. Балансування статичного навантаження також можна розглядати як детермінований розподіл завдань у системі, де перевантажений вузол з певною ймовірністю передає деякі свої завдання іншому вузлу, який не залежить від поточного стану системи. Хоча статичне балансування навантаження просто і легко аналізувати за допомогою моделей масового обслуговування, але його потенційна користь обмежена, оскільки воно не пристосовується до стану системи, що змінюється в часі. З іншого боку, динамічне рішення залежить від стану системи на момент прийняття рішення. Коли використовується динамічне вирівнювання навантаження, перевантажений вузол може передати свої завдання іншим вузлам, використовуючи інформацію про поточний стан системи. Динамічна політика за своєю суттю є більш складною, ніж будь-яка статична політика, оскільки вона вимагає, щоб кожен вузол повинен знати стани інших вузлів. Алгоритми балансування навантаження додатково поділяються на кілька кластерів відповідно до обсягу необхідної для них інформації (Shin and Chang, 1989)

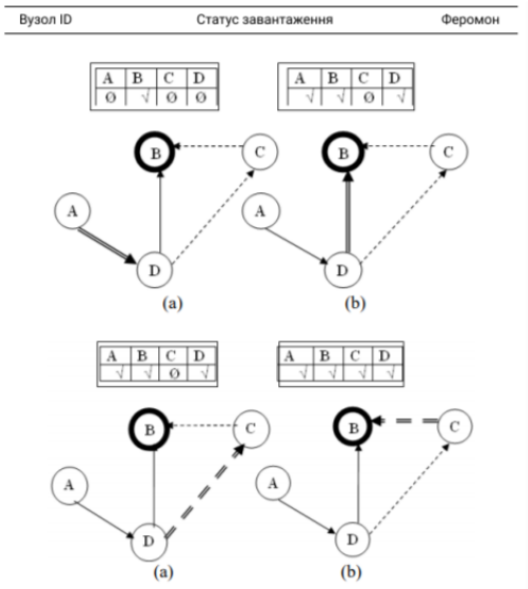
# Запропонована стратегія

У даному експерименті стратегія балансування навантаження залежить від часу та відповідає природній динаміці феромону в реальному житті. У визначений проміжок часу кожен вузол буде діяти як гніздо і посилати кількість мурах (кількість мурах залежить від стану завантаження кожного вузла; перевантажені та недозавантажені вузли відправили більше мурах). Кожен мураха буде подорожувати туром (тривалість туру залежить від розміру системи та стану завантаження вузла). Нарешті, запропонований експеримент буде порівняно зі стандартним алгоритмом викрадення роботи. Розроблена модель повністю розподілена, тобто кожен вузол вузла поводиться незалежно, а також кожен мураха чи агент, це означає, що кожен вузол або мураха є автономними. Таблиця 4.1. представляє додається інформація до кожного вузла або мурахи:

У представленій моделі кожен вузол містить інформацію про інші вузли в системі (Табл. 1.1).

У початковому стані записи в таблиці мають значення Null. У кожному мурашиному турі мураха буде нести оновлену інформацію про всі вузли, через які мураха пройшов. Після прибуття мурахи на кожен вузол будуть виконані наступні дії:

- Якщо вузол не має інформації, що міститься в мурашиній таблиці, ця інформація буде передана до таблиці вузлів без будь-якого оновлення.
- Якщо вузол містить інформацію, яка не існує в таблиці мурашок, таблиця мурашок буде оновлена.
- Якщо обидва вони мають однакову інформацію, нещодавно оновлена замінить іншу.




Таблиця 1.1 Приклад роботи моделі. (а) мураха рухається від А до Г; (б) Мураха рухається з Від D до В; (в) інший мураха рухається від D до С; (d) Мураха рухається від С до В

# Результати

Експеримент був змодельований та протестований, передбачалось, що кількість вузлів у розподіленій системі становить 30, передбачається, що мураха рухається від одного вузла до іншого за 1 часовий крок, кожне завдання передбачає 40 кроків. Для того, щоб підкреслити ефективність запропонованого алгоритму, був розглянутий випадок, коли розподілена система є дуже нерегулярною; передбачається, що вузол № 1 зайнятий 60 завданнями, а інші вузли простоюють. Рис. 4.1 показує ефективність як підходу викрадення роботи, так і підходу колонії мурах. Зрозуміло, що ефективність підходу мурашиних колоній походить від здатності розподіляти інформацію про завантаження по всіх вузлах, тур мурах був обраний випадковим чином, однак розумність мурах щодо перенесення нового стану завантаження до кожного вузла збільшується шанс кожного вузла швидко знайти хороше джерело їжі або зайнятий вузол.

# Результати

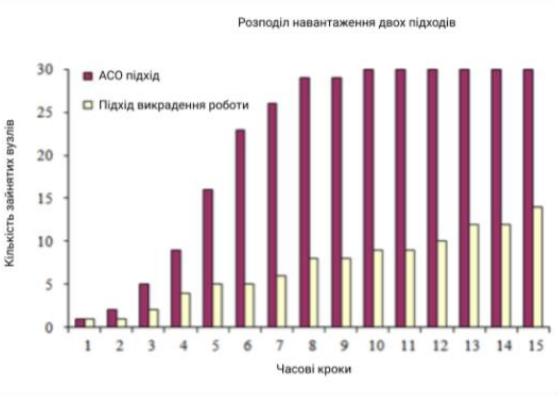



Рис. 1.1 - Розподіл навантаження: Кількість зайнятих вузлів у часових кроках

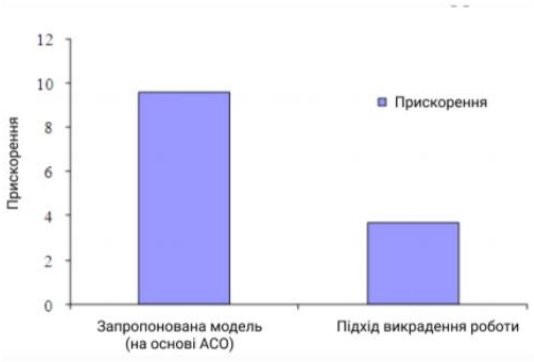



Рис. 1.2 - Порівняння прискорення запропонованої моделі та моделі викрадення роботи

# Результати

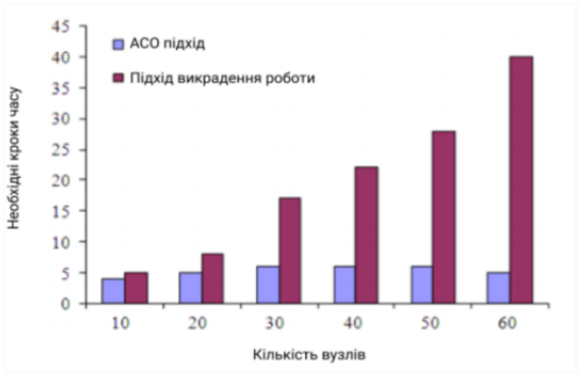



Рис. 1.3 - Динаміка балансування навантаження: Необхідні кроки часу, необхідні для досягнення розподілу навантаження 50% у порівнянні з кількістю вузлів

# Результати

Рисунок 1.2 порівнює прискорення запропонованої моделі зі стандартним підходом викрадення роботи. Виходячи з вищевказаних налаштувань, запропонована модель досягла швидкості 9,6, тоді як пришвидшення підходу - 3,7. Рисунок 1.3 вивчає ефект збільшення кількості вузлів проти кількості кроків, необхідних для підвищення розподілу навантаження до 15%. Показано, що із збільшенням кількості вузлів тимчасові кроки, необхідні для підходу до викрадення робіт, різко зростають, тоді як пропонована модель залишається майже постійною. Цей підхід та його результати дають приклад, який підкреслює важливість ройової системи в процесі прийняття рішень загалом, коли кожен агент може грати не значну роль, а глобальна поведінка може бути надійною та надійною.

# Висновки

Оптимізація колонії мурашок була і залишається плідною парадигмою для розробки ефективних алгоритмів рішення комбінаторної оптимізації. Після досліджень було продемонстровано як ефективність його застосування, так і теоретичні обгрунтування, що робить ОПМ однією з найуспішніших парадигм в метаевристичні області.

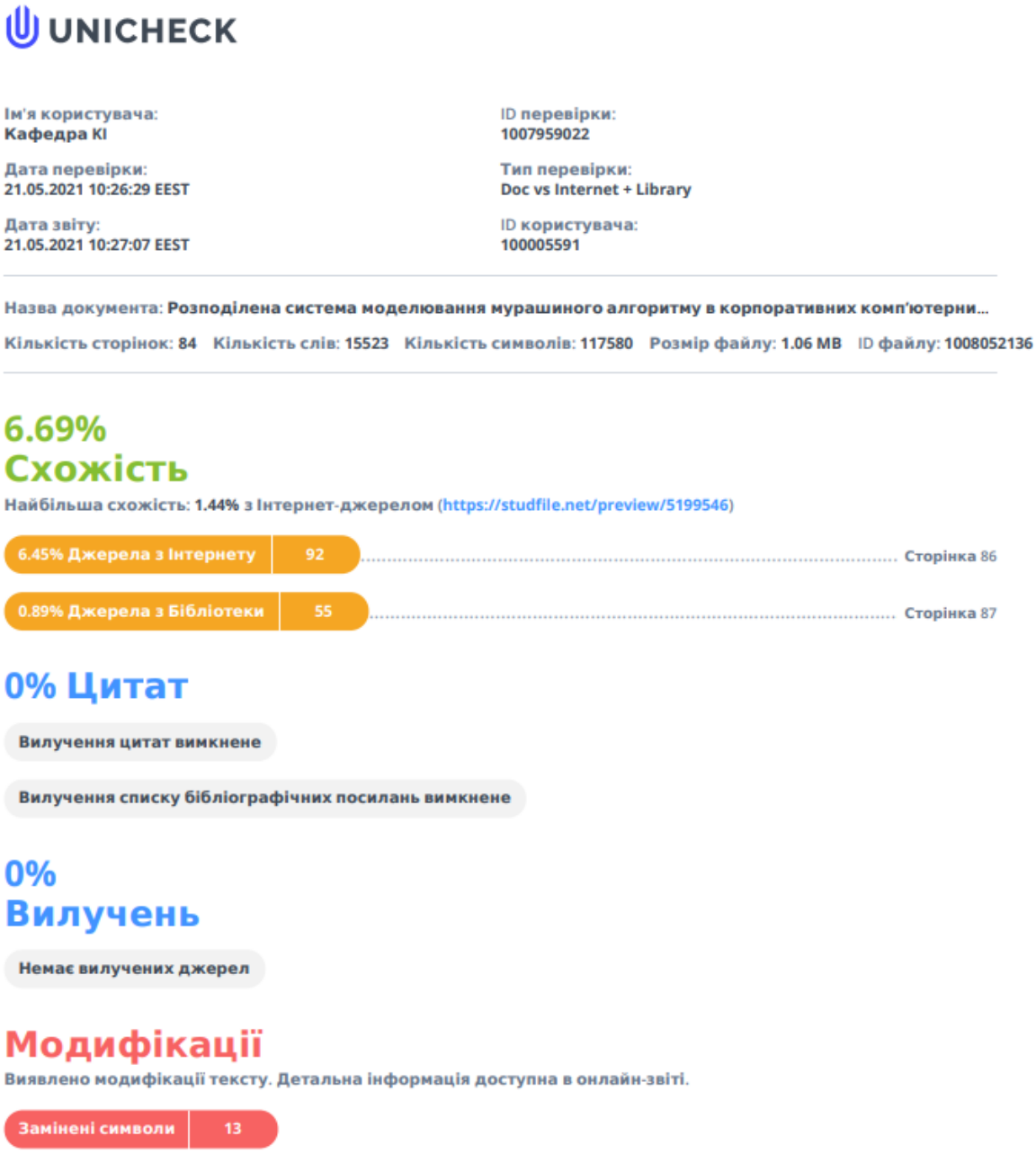
За результатами проведеного дослідження запропоновано алгоритм оптимізації корпоративної комп'ютерної мережі за допомогою мурашиних алгоритмів. Запропонований метод показав себе краще ніж стандартний алгоритм оптимізації АСО.

Першим кроком, було розглянуто Основні структурно складові корпоративної мережі, необхідне обладнання для побудови ККМ, переваги, можливості, складнощі при створенні корпоративних мереж.

Другий крок, розглянуто саму реалізації корпоративної мережі, вибір і аналіз обладнання, створення концептуальної схеми мережі. проведено мільти аналіз описані шляхи оптимізації за допомогою колоній мурашок, описані особливості і системи КМ. Описані часті проблеми оптимізації АСО а також механізми оновлення маршруту.

В результаті експеримент запропоновано підхід для балансування навантаження в розподілених системах на основі численних колоній мурашок була запропонована оптимізація. Використання кількох гнізд, або колоній мурашок у процесі пошуку, допомогло підвищити швидкість обміну інформацією по всіх вузлах системи. Крім того, динамічний обмін інформацією та її повне розповсюдження - це інші основні характеристики, що відрізняють цей підхід. Результати показали ефективність запропонованої моделі в порівнянні зі стандартним алгоритмам викрадення роботи з точки зору кількості зайнятих вузлів та витраченого часу для досягнення ефективності 15%

Результати перевірки на плагіат:



UNICHECK

Ім'я користувача: Кафедра КІ
ID перевірки: 1007959022
Дата перевірки: 21.05.2021 10:26:29 EEST
Тип перевірки: Doc vs Internet + Library
Дата звіту: 21.05.2021 10:27:07 EEST
ID користувача: 100005591

Назва документа: Розподілена система моделювання мурашиного алгоритму в корпоративних комп'ютерни...

Кількість сторінок: 84 Кількість слів: 15523 Кількість символів: 117580 Розмір файлу: 1.06 MB ID файлу: 1008052136

6.69%
Схожість

Найбільша схожість: 1.44% з Інтернет-джерелом (https://studfile.net/preview/5199546)

6.45% Джерела з Інтернету 92 ..... Сторінка 86

0.89% Джерела з Бібліотеки 55 ..... Сторінка 87

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 13

Fri May 21 09:39:25 EEST 2021, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

# Anti-Plagiarism v-15.257

**Максимальное совпадение с одним документом 3.0%**

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 7%**

| ID: 91110<br>Название: Розподілена система моделювання мурашиного алгоритму в корпоративних комп'ютерних мережах<br>Добавлено в БД: 2021-05-21<br>Авторы: Овсяк.О.В.<br>Руководители: Медзатий Д.М.<br>Консультанты:<br>Опоненты: | Документ | | Суммарное совпадение по Базе Данных | |
|---|---|---|---|---|
| | Символы | Лексемы | Символы | Лексемы |
| | 86535 | 623 | 4329 (5%) | 53 (9%) |

Источник плагиата

| ID | Описание | Наличие плагиата в документе | |
|---|---|---|---|
| | | Символы | Лексемы |

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

**РЕЦЕНЗІЯ НА ДИПЛОМНУ РОБОТУ**

Дипломник студент групи КІ2м-19-1 Овсяк О. В.

Тема Розподілена система моделювання мурашиного алгоритму в корпоративних комп'ютерних мережах

Спеціальність 123 – Комп'ютерна інженерія

**Обсяг дипломної роботи:**

Кількість листів креслень 0; кількість сторінок записки 91

1.Короткий зміст ДР та прийнятих рішень Представлена робота присвячена актуальній темі в області оптимізації розподілених систем за допомогою мурашиних колоніїі складається із наступних розділів: вступ, структурно складові мереж, реалізація компютерної корпоративної мережі, мкльтифракційний аналіз, ефективність розподілених систем та експерименти, висновки, додатки.

2. Висновок про відповідність ДР поставленому завданню Магістерська кваліфікаційна робота виконана у відповідності з завданням із дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі було розглянуто Основні структурно складові корпоративної мережі, необхідне обладнання для побудови ККМ, переваги, можливості, складнощі при створенні корпоративних мереж.У другому розділі розглянуто саму реалізації корпоративної мережі, вибір і аналіз обладнання, створення концептуальної схеми мережі.У третьому розділі проведено мільти фракційний аналіз описані шляхи оптимізації за допомогою колоній мурашок, описані особливості і системи КМ. Описані часті проблеми оптимізації АСО а також механізми оновлення маршруту. У четвертому розділі запропоновано підхід для балансування навантаження в розподілених системах на основі численних колоній мурашок була запропонована оптимізація.

4. Позитивні сторони роботи До позитивних сторін роботи слід віднести актуальність даного направлення дослідження, деталізацію аналізу усіх розглянутих стратегій вирішення проблеми та поглиблене опрацювання всіх аспектів реалізації з практичним використанням запропонованого рішення.

5. Негативні сторони роботи До негативних сторін роботи слід віднести недоліки по оформленню представленого матеріалу, що були виправлені.

6. Оцінка графічного оформлення та пояснювальної записки роботи Дані матеріали роботи є структурованими у чіткій та логічній формі та відображають послідовність виконання поставлених завдань. І хоча й в них було знайдено декілька стилістичних та орфографічних помилок, вони були пізніше усунені. Тому дане виконання пояснювальної записки та графічного оформлення заслуговує оцінки «добре».

7. Відгук про роботу в цілому Загалом, зміст представленої роботи в повній мірі розкриває обрану тему. Дослідження, проведені в матеріалах є достатньо аргументованими. Прослідковуються високі теоретичні та практичні рівні у даному виконанні. Результатом проведення досліджень стали відповідні висновки і конкретні пропозиції щодо вдосконалення процесу виявлення зловмисного програмного забезпечення у локальних комп'ютерних мережах з використанням баєсовської
мережі.

8. Інші зауваження

9. Оцінка дипломної роботи Робота заслуговує оцінки «задовільно», а її автор – присвоєння кваліфікації «магістра» з комп'ютерної інженерії.

РЕЦЕНЗЕНТ (прізвище, ім'я, по-батькові, посада, місце роботи) Кльоц Юрій Павлович, кандидат технічних наук, доцент, завідувач кафедри кібербезпеки та комп'ютерних систем та мереж

"31" травня 2021 р. _____
(підпис)

Завідувачу кафедри КІСП
д-р.техн.наук, проф. Говорущенко Т. О.

Вся Н О.В.
ПІБ здобувача вищої освіти

ФПКТС, 2 курсу, групи КІ2м-19-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіатоповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

24.05.2021
дата

підпис

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА СИСТЕМНОГО ПРОГРАМУВАННЯ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: «Розподілена система моделювання мурашинного алгоритму в корпоративних комп'ютерних мережах»

Автор: Овсяк Олександр Валентинович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-наукова

Науковий керівник: Медзатий Д. М., канд.техн.наук, доцент

Після аналізу звіту подібності зроблено такий висновок:

| № | Висновок | Позначка про відповідність |
|---|---|---|
| 1 | Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту. | відповідає |
| 2 | Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи | |
| 3 | Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат. | |
| 4 | Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту. | |

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) в тексті кваліфікаційної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень в бланках (титулка, бланк завдання, в структурі підрозділів ВСТУПУ) та в назвах публікацій джерел посилання;
2) найбільшу схожість встановлено з одним документом і становить вона 1.44 відсотка в частині загальноприйнятої термінології;
3) збігів та ідентичності в тексті кваліфікаційної роботи немає, наявна лише схожість.

Сумарний обсяг всіх запозичень, визначений системою виявлення схожості, складає 6.69% і адресується до 433 першоджерела, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Д. М. Медзатий

Професор кафедри КІСП _____ С. М. Лисенко

Завідувач кафедри КІСП _____ Т. О. Говорущенко