МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

НН Інститут комп'ютерних наук та управління проектами

(повне найменування інституту, назва факультету)

Програмного забезпечення автоматизованих систем

(повна назва кафедри)

Пояснювальна записка

до магістерської роботи за темою

Розробка моделі розподілення навантажень інформаційно- вимірювальних систем за допомогою використання генетичного алгоритму

Виконала: студентка 6 курсу, групи_6153мз

напряму підготовки (спеціальності)

152 «Метрологія та
інформаційно-вимірювальна техніка»

_____Тарасова А. О.

Завідувач кафедри___ПЗАС, д.т.н., проф.__

_____Приходько С.Б.

Керівник МР ____к.т.н., доцент,

_____Пономаренко Т.В.

м. Миколаїв – 2020 р.

Міністерство освіти і науки України
Національний університет кораблебудування
імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Освітній ступень Магістр
Галузь 12 «Інформаційні технології»
(шифр і назва)
Спеціальність 121 «Інженерія програмного забезпечення»
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

" 26 " 10 2020 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ (МАГІСТЕРСЬКУ) РОБОТУ СТУДЕНТУ

(прізвище, ім'я, по батькові)

1.Тема магістерської роботи
керівник роботи
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " 26 " жовтня 2020 року №1037-уч
2. Строк подання студентом роботи 01.12.2020 року
3. Вихідні дані до роботи
4.Зміст магістерської роботи (МР):
• Титульний аркуш, завдання на кваліфікаційну (магістерську) роботу, реферат (українською, англійською), зміст, перелік умовних позначень, символів, одиниць та термінів (за необхідності).
• Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень. Публікації.)
• Огляд літератури за темою, обґрунтування необхідності проведення досліджень за обраною темою, вибір напрямків досліджень, мета дослідження, основні задачі дослідження
• Викладення результатів власних досліджень з висвітленням того нового, що пропонується
• Проект програмного забезпечення
• Результати досліджень та розробки проекту програмного забезпечення•

РЕФЕРАТ

Тарасової Анастасії Олегівни

«Розробка моделі розподілення навантажень інформаційно-вимірювальних систем за допомогою використання генетичного алгоритму»

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2020 р.

Обсяг роботи: 100 стор., 11 табл., 17 рис., 14 використаних джерел, 5 додатків.

Актуальність теми роботи: необхідність підвищення ефективності розподілення навантажень інформаційно-вимірювальних систем.

Мета та завдання дослідження. Метою є підвищення швидкості обробки запитів задяки розподіленню навантажень на складові інформаційно-вимірювальної системи за допомогою використання генетичного алгоритму.

Об'єктом дослідження є процес розподілення навантажень інформаційно-вимірювальних систем.

Предметом дослідження є модель розподілення навантажень інформаційно-вимірювальних систем за допомогою використання генетичного алгоритму.

Методи дослідження. У вирішенні поставлених завдань використано методи теорії прийняття рішень, теорія ймовірності, математичної статистики,оптимізаційні методи.

Наукова новизна одержаних результатів. Розроблено модель розподілення навантажень інформаційно-вимірювальних систем, яка за допомогою використання генетичного алгоритму дає можливість мінімізувати максимально навантаження в мережі інформаційно-вимірювальної системи.

Практичне значення одержаних результатів. Розроблено програму для розподілення навантажень інформаційно-вимірювальної системи .

Апробація результатів роботи. Робота пройшла апробацію на конференції Комп’ютерна інженерія і кібербезпека : досягнення та інновації : матеріали II Всеукр. наук.-практ. конф. здобувачів вищої освіти й молодих учених, м. Кропивницький, 25–27 листоп. 2020 р. .

Публікації. За результатами опубліковано тези у збірнику матеріалів конференції Комп’ютерна інженерія і кібербезпека : досягнення та інновації : матеріали II Всеукр. наук.-практ. конф. здобувачів вищої освіти й молодих учених, м. Кропивницький, 25–27 листоп. 2020 р.

Ключові слова: МОДЕЛЬ РОЗПОДІЛЕННЯ НАВАНТАЖЕНЬ, ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНА СИСТЕМА, ГЕНЕТИЧНИЙ АЛГОРИТМ.

ABSTRACT

Of Tarasova Anastasia Olehivna

“Development of the model of load distribution of information-measuring systems by means of use of genetic algorithm”

The qualification work in obtaining a master's degree in specialty 152 - "Metrology and information-measuring equipment". Admiral Makarov National University of Shipbuilding. Mykolayiv, 2020.

The qualification work is presented on the 100 pages of typewritten text, contains 11 tables, 17 figures, 5 appendices and 14 references.

"Development of a model of load distribution of information and measuring systems using a genetic algorithm "

Relevance of the topic: the need to increase the efficiency of load distribution of information and measuring systems.

The goal and objectives of the study. The goal of the study is to increase the efficiency of load distribution of information and measurement systems through the use of a genetic algorithm.

Objectives of the study: to analyze existing models for load distribution of information and measuring systems; develop a load balancing program for the EnergyDB information and measurement system.

The object of study is the process of load distribution of information and measuring systems.

The subject of the study is a model of load distribution of information and measuring systems using a genetic algorithm.

The methods of the study. The methods of probability theory, of decision theory and mathematical statistics, regression analysis are used in solving the problems.

The scientific novelty of obtained results. A model of load distribution of information and measuring systems using a genetic algorithm has been developed, which in comparison with the model based on the model of a closed network of queuing systems has a higher percentage of predicted results, less than the average relative error.

The practical significance of the obtained results. A program for load balancing of the EnergyDB information and measurement system has been developed.

Approbation of research results. The research results were published at the All-Ukrainian scientific-practical Internet conference.

Publications. The results of the work are presented in 1 publication, namely: 1 materials of the All-Ukrainian scientific-practical Internet conference.

Keywords: LOAD DISTRIBUTION MODEL, INFORMATION-MEASURING SYSTEM, GENETIC ALGORITHM.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

ГА — Генетичний алгоритм — це еволюційний алгоритм пошуку, що використовується для вирішення задач оптимізації і моделювання, шляхом послідовного підбору, комбінування і варіації шуканих параметрів з використанням механізмів, що нагадують біологічну еволюцію.[1]

Ген – реально існуюча, незалежна одиниця спадковості, що комбінується й розщеплюється при схрещуваннях;

Хромосома – структурний елемент клітинного ядра біологічних організмів, який є носієм генів у клітинному ядрі особини.

Популяція – досить велике співтовариство організмів, що схрещуються між собою.

Селекція – це вибір тих хромосом, які будуть брати участь в створенні нащадків для наступної популяції, тобто для чергового покоління.

Схрещування – генетичний оператор, що відповідає за передачу ознак від батьків до потомків.

Мутація – генетичний оператор, що відповідає за зміну генів особини в допустимих значеннях для покращення його цільової функції.

Фітнес-функція – це функція оцінки генетичного алгоритму, що визначає міру пристосованості отриманого рішення.

ЗМІСТ

ВСТУП

Розподілення навантаження каналів зв'язку стало важливим компонентом для побудови та оптимізації навантаження на компоненти інформаційно-вимірювальних систем, зібраних як на базі декількох Raspberry Pi 3, так і подібних мікропроцесорних вимірювальних систем. У сучасних мережах існує потреба в оптимізації рішення щодо передачі даних (трафіку) за багатьма критеріями. Класичні алгоритми не застосовуються до задач цього типу, що належать до класу NP-тяжких. Виникає необхідність у формуванні нових підходів та алгоритмів розподілення навантаження на канали зв'язку в мережі.

Парадигму генетичних алгоритмів запропонував Джон Холланд, що опублікував на початку 60-х років її основні положення. А загальне визнання вона одержала після виходу у світ в 1975 році його класичної праці «Адаптація в природних і штучних системах». Генетичний алгоритм був отриманий у процесі узагальнення й імітації в штучних системах таких властивостей живої природи, як природний відбір, пристосованість до мінливих умов середовища, спадкування нащадками життєво важливих властивостей від батьків і т.д. Тому що алгоритми в процесі пошуку використовують деяке кодування множини параметрів замість самих параметрів, то він ефективно застосовуються для рішення дискретних і комбінаторних задач оптимізації, визначених як на числових множинах, так і на кінцевих множинах довільної природи. Оскільки для роботи алгоритму в якості інформації про функцію, що оптимізується використовуються лише її значення в розглянутих точках простору пошуку, і не потрібно обчислень ні похідних, ні яких-небудь інших характеристик, то даний алгоритм може бути застосований до широкого класу функцій, зокрема, тих що не мають аналітичного опису. Використання набору початкових точок дозволяє застосовувати для їх формування різні способи, що залежать від специфіки розв'язуваної оптимізаційної задачі, у тому числі можливе задання

такого набору безпосередньо людиною. Тобто в цьому випадку застосовується єдине універсальне представлення будь-якої задачі [1]

Мета та завдання дослідження. Метою є підвищення ефективності розподілення навантажень інформаційно-вимірювальної системи за допомогою використання генетичного алгоритму.

Завдання дослідження: проаналізувати існуючі моделі для розподілення навантажень інформаційно-вимірювальних систем; розробити програму розподілення навантаження інформаційно-вимірювальної системи EnergyDB.

Об'єктом дослідження є процес розподілення навантажень інформаційно-вимірювальних систем.

Предметом дослідження є модель розподілення навантажень інформаційно-вимірювальних систем за допомогою використання генетичного алгоритму.

Методи дослідження. У вирішенні поставлених завдань використано методи теорії прийняття рішень, теорію ймовірності, математичної статистики, евристичні методи.

Наукова новизна одержаних результатів. Розроблено модель розподілення навантажень інформаційно-вимірювальних систем, яка за допомогою використання генетичного алгоритму дає можливість минімізувати максимально навантаження в мережі інформаційно-вимірювальної системи.

Практичне значення одержаних результатів. Розроблено програму для розподілення навантажень інформаційно-вимірювальної системи EnergyDB.

Апробація результатів досліджень. Результати досліджень були оприлюднені на у збірнику матеріалів конференції Комп’ютерна інженерія і кібербезпека : досягнення та інновації : матеріали II Всеукр. наук.-практ. конф.

здобувачів вищої освіти й молодих учених, м. Кропивницький, 25–27 листоп. 2020 р

Практичне значення отриманих результатів полягає в розробці програмного забезпечення яке використовує генетичний алгоритм для Розподілення навантаження та розподілення навантаження між складовими інформаційно-вимірювальних систем, що на відміну від системи на основі моделі замкненої мережі СМО показує більшу ефективність.

Особистий внесок здобувача. Проведені дослідження, зібрані емпіричні дані та отримані результати, наведені в магістерській роботі, одержані здобувачем особисто.

Публікації. За результатами опубліковано тези у збірнику матеріалів конференції Комп’ютерна інженерія і кібербезпека : досягнення та інновації : матеріали II Всеукр. наук.-практ. конф. здобувачів вищої освіти й молодих учених, м. Кропивницький, 25–27 листоп. 2020 р.

1. АНАЛІЗ ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНОЇ СИСТЕМИ

Інформаційно-вимірювальна система, що є предметом дослідження створена для енергетичного менеджменту для розумних логерів та лічильників та сенсорів об’єднаних в одну інформаційну мережу, що встановлені на підприємствах. Вона має багато складових, що входять до її складу і розподілення навантаження між ними з метою оптимізації таблиці маршрутизації є актуальним завданням. На рисунку1 зображено складові досліджуваної інфор маційно-вимірювальної системи.

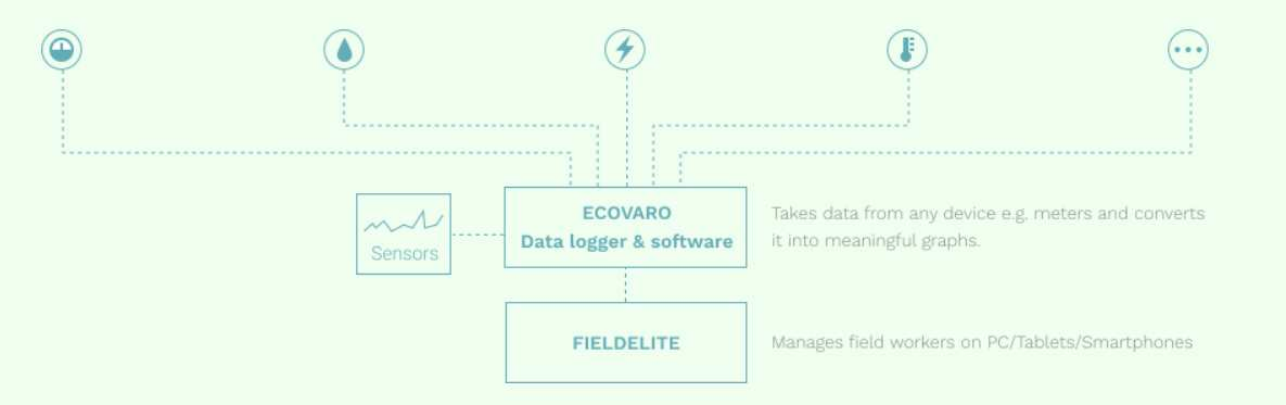

Рисунок 1.1 – Складові інформаційно-вимірювальної системи

Більшість розумних пристроїв що під’єднані до системи мають мінікомп’ютерну складову, що дає можливість налаштувань їх для комунікації між собою різними шляхами.

Система займається збором, аналізом та контролем обсягів різних типів енергії на підприємствах де вона встановлена. В рамках кваліфікаційної роботи мені було надано доступ на тестовий сервер компанії-власника цієї системи, але все одно я підписувала угоду про нерозголошення даних, тому всі зображення наводжу з тестовими даними.

Використання генетичних алгоритмів для різного роду практичних задач з ростом розрахункових потужностей останнім часом набуває великої

популярності – так в роботах 1-6 з допомогою генетичних алгоритмів вирішено практичні задачі створення розкладу занять, об’єднання комп’ютери у кластери, вирішення задач оптимального плану розкрою.

Силою генетичних алгоритмів є маніпулювання одночасно багатьма параметрами, тому застосування цього типу оптимізаційних методів у інформаційно-вимірювальній системі, що є дуже розгалуженою і багатоскладовою є актуальним завданням. Найкраще генетичні алгоритмі працюють у комплексі з класичними методами, що дає можливість підвищити їх ефективність

Генетичний алгоритм складається з етапів наведених на рисунку 1.2.

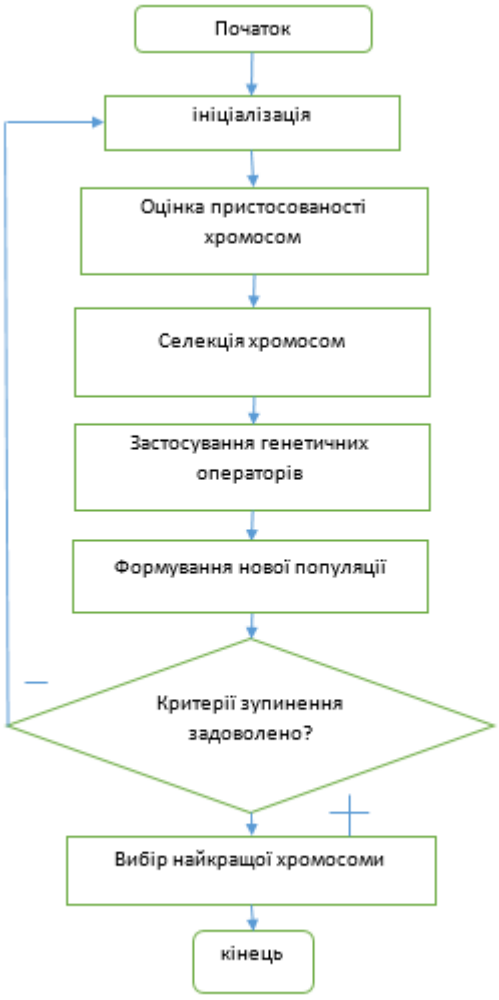

Рисунок 1.2 Блок- схема роботи генетичного алгоритму

Основні джерела інформації, що необхідні для вирішення заданої задачі, це джерела з описом генетичних алгоритмів та їх застосуванням у прикладних задачах [1].

Основна мета схрещування – отримання нових варіантів розв'язків порівняно з тими що вже існують. Тому обов'язковою умовою є те, що внаслідок схрещування двох батьківських особин повинні виникнути коректні умови відбору нащадків, мета розробки моделі є нагальною [2]

Алгоритми розподілення навантаження поділяються на статичні, динамічні і адаптивні. Динамічні алгоритми розподілу навантаження використовують інформацію про стан системи, наприклад, про завантаженість вузлів, для прийняття рішень про розподіл навантаження в процесі роботи системи. Статичні алгоритми подібної інформації не використовують. У статичних алгоритмах рішення про розподіл навантаження реалізовані заздалегідь, до початку роботи системи, і не змінюються в процесі роботи. Прикладом статичного алгоритму може служити алгоритм Round Robin, що розподіляє завдання між вузлами по черзі [2].

Динамічні алгоритми завдяки використанню інформації про стан системи мають можливість обігнати по ефективності статичні алгоритми, наприклад, простий динамічний алгоритм може бути ідентичний за паливною ефективністю складного статичному. Внаслідок того, що динамічно алгоритми повинні збирати, зберігати і аналізувати інформацію про стан системи, їм властиві великі накладні витрати на балансування в порівнянні зі статичними алгоритмами.

Адаптивні алгоритми є окремим випадком динамічних алгоритмів і в залежності від умов і апріорного знання про властивості алгоритмів балансування вибирають найбільш підходящий з них [3].

Інформація про стан системи є важливим елементом роботи динамічного алгоритму. Існують різні методи збору даних про завантаженість системи і

побудови характеристик завантаженості. Завантаженість вузла може визначатися, наприклад, як частка процесорного часу, витрачена на вирішення завдання, як кількість завдань в черзі вузла, або як ймовірність виявити вузол зайнятим вирішенням завдання в будь-який момент часу.

По відношенню до моменту початку передачі завдань розрізняють витісняючі і невитисняючі алгоритми балансування. Витісняючі алгоритми можуть передавати завдання, вирішені частково, в той час, ті що не витісняють можуть розподіляти тільки завдання, виконання яких ще не розпочато. Прикладом витісняючого алгоритму може служити алгоритм планування процесів в системі Unix.

Динамічні алгоритми балансування розрізняються ступенем централізованості. Алгоритми можуть бути централізованими, ієрархічними, повністю розподіленими або поєднувати зазначені підходи. Централізовані алгоритми потенційно менш надійні, ніж повністю розподілені, тощо збій центрального компонента призведе до збою всієї системи. Рішенням даної проблеми може служити підтримання надлишкового числа компонентів, які дублюють один одного. Інша слабкість централізованих алгоритмів полягає в тому, що центральний компонент являє собою можливе вузьке місце системи. У той час як ієрархічні алгоритми дозволяють частково вирішити обидві проблеми, повне рішення досягається тільки розподіленими алгоритмами. З іншого боку, централізовані алгоритми простіше контролювати і для них простіше виконувати налагодження.

Механізм збору інформації про завантаженість системи визначає, коли збираються дані про завантаженість, що є джерелом інформації і де інформація зберігається. Виділяють кілька класів механізмів збору інформації.

1) збір даних по необхідності. розподілені алгоритми даного класу збирають інформацію про завантаженість, коли вузол потребує балансуванню навантаження. вони підрозділяються на ініційовані відправником, одержувачем і симетрично ініційовані. в алгоритмах,

ініційованих відправником, вузол, що передає завдання, шукає одержувачів, яким він може передати частину завдань. в алгоритмах, ініційованих одержувачем, вузол, який отримує завдання, запозичує завдання у відправника. у симетричних алгоритмах використовується комбінація згаданих підходів.

2) періодичний збір даних. алгоритми даного класу можуть бути як централізованими, так і розподіленими. залежно від зібраних даних, алгоритм ініціює балансування навантаження.

3) збір даних щодо зміни стану. у системах, які реалізують алгоритми даного класу, вузли самі поширюють інформацію про зміну завантаженості при зміні внутрішнього стану. У разі розподілених алгоритмів дані направляються сусіднім вузлам, в разі централізованих алгоритмів дані направляються координатору. Під стабільністю алгоритму балансування або системи, його використовує, зазвичай розуміють одну з двох характеристик:

- системна стійкість, тобто неприпустимість ситуації, коли окремі вузли системи перевантажені, в той час як інші простоюють або недовантажені;
- алгоритмічна стабільність, виражена в тому, що алгоритм не робить непотрібних дій з ненульовий ймовірністю, наприклад, не передає завдання по колу між вузлами так, що завдання виконане буде ніколи.

Класичні методи оптимізації, як правило, не відповідають визначенню глобального мінімуму багатоекстремальних функцій, тому в ряді робіт було запропоновано використовувати для рішень багатоекстремальних завдань так звані гібридні генетичні алгоритми [1, 2]. Їх суть складається у спільному використанні генетичного алгоритму та деякого класичного методу оптимізації. Генетичний алгоритм на етапі своєї роботи забезпечує ефективне зменшення

простору пошуку, класичний метод оптимізації у своїй черговій високій швидкості ходу поблизу точок екстремума. [3]

Генетичні алгоритми (ГА) працюють за аналогією з природою. Вони оперують з сукупністю “особин”, що представляють собою рядки, кожна з яких кодує одне з рішень задачі [1–3]. Пристосованість особини оцінюється за допомогою спеціальної функції. Найбільш пристосовані отримують шанс схрещуватися і давати потомство. Найгірші особини видаляються і не беруть участі в подальшому розвитку (еволюції). Методи з комбінованим використанням ГА, мають більшу ефективність вирішення певного класу задач оптимізації, ніж просто використання класичних методів. Постановка задачі В даний час ГА використовуються для вирішення наступних задач: • пошук глобального екстремуму багатопараметричної функції; • оптимізація функцій; • оптимізація динамічних систем; • задачі оптимізації параметрів; • настройка штучної нейронної мережі; • завдання на графах (задача комівояжера); • навчання інтелектуальних моделей; • комбінаторні задачі; • знаходження коренів систем нелінійних рівнянь.[5]

1.2 Постановка задачі дослідження та основних вимог до розробки програмного забезпечення розподілення навантаження інформаційно-вимірювальних систем з використанням генетичного алгоритму

У числі цілей, для досягнення яких використовується розподілення навантаження, потрібно виділити наступні:

- справедливість: потрібно гарантувати, щоб на обробку кожного запиту виділялися системні ресурси і не допустити виникнення ситуацій, коли один запит обробляється, а всі інші чекають своєї черги;
- ефективність: всі сервери, які обробляють запити, повинні бути зайняті на 100%; бажано не допускати ситуації, коли один з серверів простоює в

очікуванні запитів на обробку (відразу ж обмовимося, що в реальній практиці ця мета досягається далеко не завжди);

- скорочення часу виконання запиту: потрібно забезпечити мінімальний час між початком обробки запиту (або його постановкою в чергу на обробку) і його завершення;
- скорочення часу відгуку: потрібно мінімізувати час відповіді на запит користувача.

Дуже бажано також, щоб алгоритм розподілення навантаження мав такі властивості:

- передбачуваність: потрібно чітко розуміти, в яких ситуаціях і при яких навантаженнях алгоритм буде ефективним для вирішення поставлених завдань;
- рівномірна завантаження ресурсів системи;
- масштабуємось: алгоритм повинен зберігати працездатність при збільшенні навантаження.

Більш докладно вимоги до програмного забезпечення викладено в технічному завданні в додатку А.

2 РОЗРОБКА ЙМОВІРНІСНОЇ МОДЕЛІ РОЗПОДІЛЕННЯ НАВАНТАЖЕННЯ ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНИХ СИСТЕМ З ВИКОРИСТАННЯМ ГЕНЕТИЧНОГО АЛГОРИТМУ.

У задачі розподілення навантаження інформаційно-вимірювальна система представляється у вигляді зваженого орієнтованого графа $G = (V, E, C)$, де V - безліч вершин, – множина ребер графа, $C_{i\ j}$: E -> R - пропускна здатність кожного з'єднання (ребра графа). Інформація про потік між кожною парою вершин задається функцією F_{ij}, де i – вихідна вершина, j – вершина призначення. Запропонований алгоритм може бути застосований і для окремої підсистеми. В такому випадку використовується певний підграф графа G.

Генетичний алгоритм розподілення навантаження використовує таблицю маршрутизації мережі, що містить $m_{i,j}$ альтернативних шляхів, між кожною парою вершин (i, j). ця таблиця зберігає всі проміжні вершини шляху (I –> j), а також «ціну» кожного маршруту виходячи з алгоритму маршрутизації. Такий підхід дозволяє застосовувати генетичний алгоритм в складових інформаційно-вимірювальних систем, використовуючи маршрути між двома вершинами з різною метрикою і адміністративною дистанцією.

Під час функціонування мережі кожен маршрут між вершинами i та j використовується з певною ймовірністю k (i, j), причому

$$\sum_k \alpha_k(i,j) = 1,\ k = 1 \ldots m_{i,j}. \tag{1}$$

Ступінь використання пропускної здобності звязку між серверами можна вирахувати наступним чином

де

$$X_{l,m}^{k}(i,j)=\begin{cases}1, \text{ если } (l,m)\in(i\to j)_k\\ 0, \text{ если } (l,m)\notin(i\to j)_k\end{cases}. \quad (2)$$

Нехай тоді задача розподілення навантаження зводиться до мінімізації найбільшого споживання каналу передачі даних тобто до пошуку такої конфігурації маршрутів, для якої буде найменшим

$$\Phi=\min\left(P_{\max}\right). \quad (3)$$

2.1 Генетичний алгоритм

Хромосома певного покоління генетичного алгоритму *Y* складається з послідовності алелей *Ai,j*, які в свою чергу містять набори коефіцієнтів для кожної пари вершин (*i, j*), тобто

$$A_{i,j}=\left\{\alpha_k(i,j)\right\}. \quad (4)$$

Множина складається з усіх пар вершин, для яких застосовується алгоритм розподілення навантаження. Для підвищення ефективності роботи генетичного алгоритму та зменшення витрат на обчислення доцільно також обмежити множину зв'язків мережі (ребер графа *G*), для яких застосовується генетичний алгоритм. Позначимо цю множину .

Таке спрощення дозволяє виділити критичні елементи мережі і значно поліпшити динамічні характеристики алгоритму. Запропонована модель

забезпечує однакову кількість і розташування алелей в хромосомах довільного покоління (рис. 2.1).

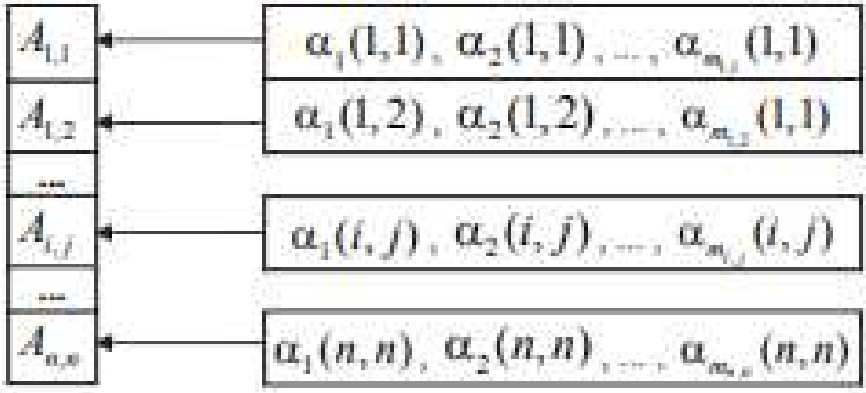

Рисунок 2.1 – Розташування алелей у хромосомі

2.1.1 Початкова популяція

Нехай для кожної пари вершин (i, j) існує альтернативних маршрутів . Користуючись метрикою алгоритму маршрутизації, кожному маршруту $_k$ ставиться у відповідність певна величина , що характеризує ціну цього шляху з урахуванням алгоритму маршрутизації. При застосуванні декількох алгоритмів маршрутизації ця величина залежить від метрики маршруту і адміністративної дистанції протоколу маршрутизації. Коефіцієнти початкової популяції генетичного алгоритму вибираються так:

$$\alpha_k(i,j) = \frac{Q_{i,j}^k}{\sum_k Q_{i,j}^k}. \tag{5}$$

Слід зауважити, що, застосовуючи алгоритм, доречно використовувати додаткові умови для співвідношення коефіцієнтів $\alpha_k(i,j)$:

$$\Omega(\alpha_k(i,j)). \quad (6)$$

Запропоновані додаткові умови накладають обмеження на відношення величин потоків даних для різних маршрутів між компонентами. Однак такі обмеження необхідні для зменшення можливого негативного впливу алгоритму Розподілення навантаження на динаміку передачі даних між певними точками в мережі.

2.1.2 Мутація і кросовер

Операція мутації для запропонованого алгоритму полягає в випадковій зміні певного алеля хромосоми *Ai, j* (одноточкова мутація) або зміні декількох алелей хромосоми одночасно (багатоточкова мутація). Операція мутації алелів в хромосомі має відповідати умовам (1) та (6).

Для вибору точок мутації використовується евристичний підхід. Оскільки завдання алгоритму полягає в мінімізації найбільшого споживання каналів, то лише зміна величини навантаження на канал з найбільшим споживанням може поліпшити рішення задачі в наступному поколінні генетичного алгоритму.

Нехай в межах поточного покоління максимальне споживання каналу спостерігається для каналу (i, j) (l, m) . отже мутації підлягають тільки ті аллелі *Ai,j*, для яких: $\exists k \in (1 \ldots m_{i,j})$, таке, що $\alpha_k(i,j) \neq 0$ и $X_{l,m}^k(i,j) = 1$. (7)

Аналогічно вибираються і точки кросовера. В силу запропонованого способу представлення рішення довжина хромосоми залишиться постійною, тому в описаному алгоритмі операція кросовера полягає в обміні частинами або окремими алелями батьківських хромосом [5].

При формуванні операцій кросовера і мутації слід враховувати два аспекти:

1. збереження послідовностей алелей, відповідних позитивному приросту функції пристосованості потрібно для забезпечення збіжності алгоритму в

цілому;

2. ефективне створення нових послідовностей алелей для забезпечення різноманітності хромосом в межах покоління і уникнення збіжності до локального екстремуму цільової функції.

Забезпечення описаних вимог здійснюється шляхом адаптивної зміни ймовірностей операцій кросовера і мутації хромосом генетичного алгоритму [6].

Визначимо ймовірності кроссовера і мутації довільної хромосоми *Y*:

$$p_c = \begin{cases} p_1 \left(F_{\max} - F(Y)\right) / \left(F_{\max} - F(Y)\right), F(Y) \geq \overline{F} \\ p_2, \; F(Y) < \overline{F} \end{cases} . \quad (8)$$

$$p_m = \begin{cases} p_3 \left(F_{\max} - F(Y)\right) / \left(F_{\max} - F(Y)\right), F(Y) \geq \overline{F} \\ p_4, \; F(Y) < \overline{F} \end{cases} . \quad (9)$$

де - значення функції пристосованості кращої хромосоми популяції, F – середнє значення функції пристосованості хромосом в популяції $0 < p1, p2, p3, p4 \leq 1$.

Для забезпечення різноманітності хромосом в межах популяції вірогідність кросовера і мутації найменш пристосованих хромосом повинна бути високою, тому в запропонованому алгоритмі p1 = p2 = 1, p3 = p4 = 0,5.

Такий адаптивний підхід до операцій кросовера і мутації забезпечує еволюцію популяції і покращує збіжність алгоритму. З іншого боку, при зростанні кількості однакових хромосом в популяції ймовірності мутації і кросовера ростуть. Таким чином, запропонований алгоритм дозволяє уникнути передчасної збіжності до локального екстремуму.

2.2.3 Відбір

Оскільки рівність (3) відповідає завданню пошуку мінімуму, функція пристосованості хромосоми *Y* може бути обчислена як:

$$F(Y) = \left(\max_{(l,m) \in E'} (P(l,m)) \right)^{-1} . \quad (10)$$

Алгоритм використовує пропорційний метод відбору в поєднанні з методом збереження кращих особин популяції (елітизмом). Значить, ймовірність відбору певної хромосоми пропорційна значенню функції пристосованості цієї хромосоми. Нехай розмір популяції *N*, а значення функції пристосованості хромосоми *Yi*, так само *F (Yi*), тоді ймовірність відбору *Yi* обчислюється так:

$$p_i^s = F(Y_i) \cdot \left(\sum_i F(Y_i) \right)^{-1} . \quad (11)$$

Оскільки при описаному підході існує ймовірність статистичної помилки і втрати кращих хромосом, в запропонованому алгоритмі застосовується метод збереження кращих рішень. При цьому гарантується перехід кращих особин наступного покоління генетичного алгоритму і збіжність алгоритму в цілому.

2.5 Маршрутизація

Після завершення роботи генетичного алгоритму Розподілення навантаження його результати повинні бути використані в процесі маршрутизації мережі. В сучасних протоколах процес маршрутизації потоку даних між вершинами *i* та |*j* на проміжних вершинах базується на інформації про вершині призначення *j* пакета даних, який проходить через проміжну вершину.

Оскільки алгоритм Розподілення навантаження базується на інформації про всіх проміжних вершинах маршрутів, слід розширити класичний формат таблиці маршрутизації і алгоритму маршрутизації в цілому.

Розглянемо таблицю маршрутизації на проміжній вершині *k* мережі. У пропонованому підході кожний запис таблиці, відповідний маршруту, який проходить через вершину *k*, додатково містить інформацію про початкову вершину *i*. Якщо існує декілька альтернативних шляхів між вершинами *i* та *j*, то запис також містить ймовірність вибору певного маршруту.

Нехай існує *m* шляхів від *i* до *j*, що проходять через вершину *k*, тоді ймовірність вибору деякого маршруту визначається так:

$$\beta_s(i,j) = \frac{\alpha_s(i,j)}{\sum_s \alpha_s(i,j)}, \quad s = 1 \ldots m . \tag{12}$$

Таким чином, результати роботи генетичного алгоритму Розподілення навантаження в мережі інтегрується в додаткові записи таблиці маршрутизації в кожній проміжній вершині обміну трафіком.

Алгоритм маршрутизації пакета даних в даному випадку виглядає так:

– пошук набору маршрутів, відповідних інформації про вершині призначення пакета даних;

– відбір маршрутів, відповідних інформації про початкову вершину;

– застосування визначеного маршруту на основі ймовірності його вибору.

Існуючі записи таблиці не змінюються, забезпечуючи роботу системи за відсутності алгоритму розподілення навантаження і в разі, коли проміжна вершина або певний маршрут не використовуються алгоритмом розподілення навантаження.

Висновок за розділом 2.

Описаний підхід застосовується для побудови та оптимізації навантаження на компоненти інформаційно-вимірювальних систем, зібраних як на базі декількох Raspberry Pi 3, так і подібних мікропроцесорних вимірювальних систем. Його реалізація зробить можливим ефективніше використовувати ресурси та значно покращити динамічні характеристики систем за зменшити навантаження складових інформаційно-вимірювальної системи вцілому.

3 ПРОЄКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗПОДІЛЕННЯ НАВАНТАЖЕННЯ ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНОЇ СИСТЕМИ З ВИКОРИСТАННЯМ ГЕНЕТИЧНОГО АЛГОРИТМУ

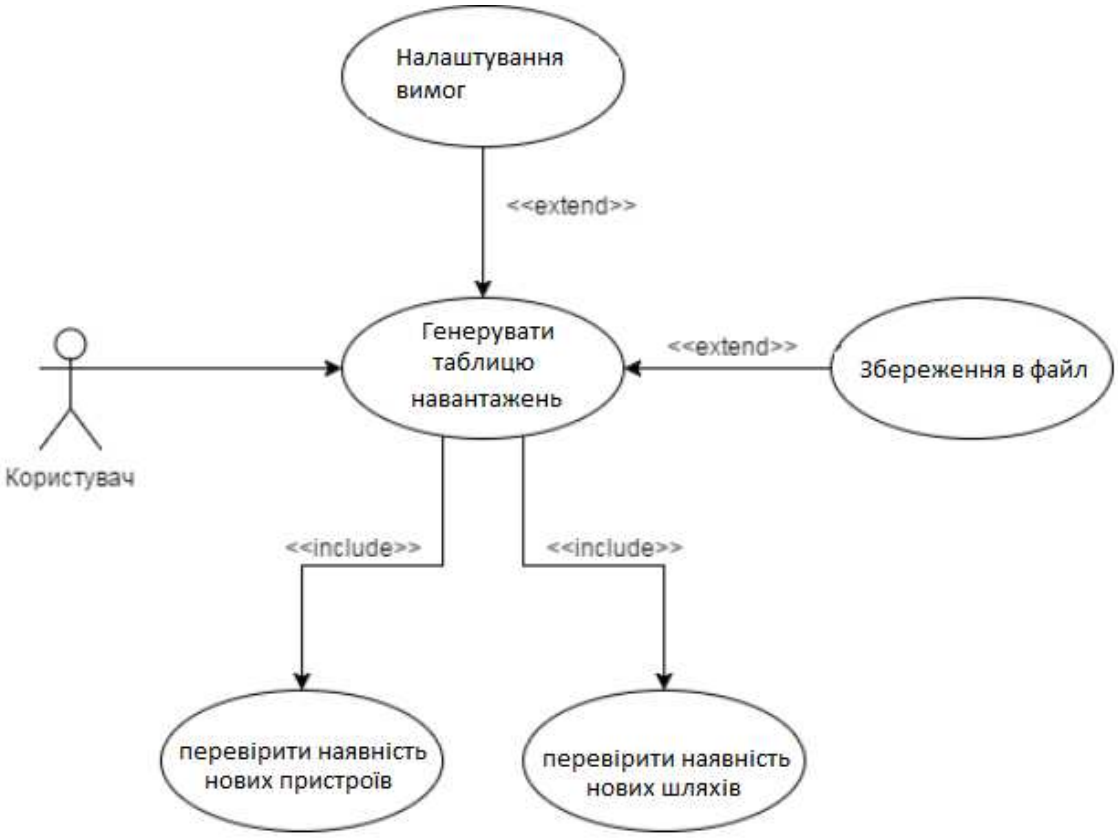

Рисунок 3.1 – Діаграма Use Case для програмного забезпечення розподілення навантаження

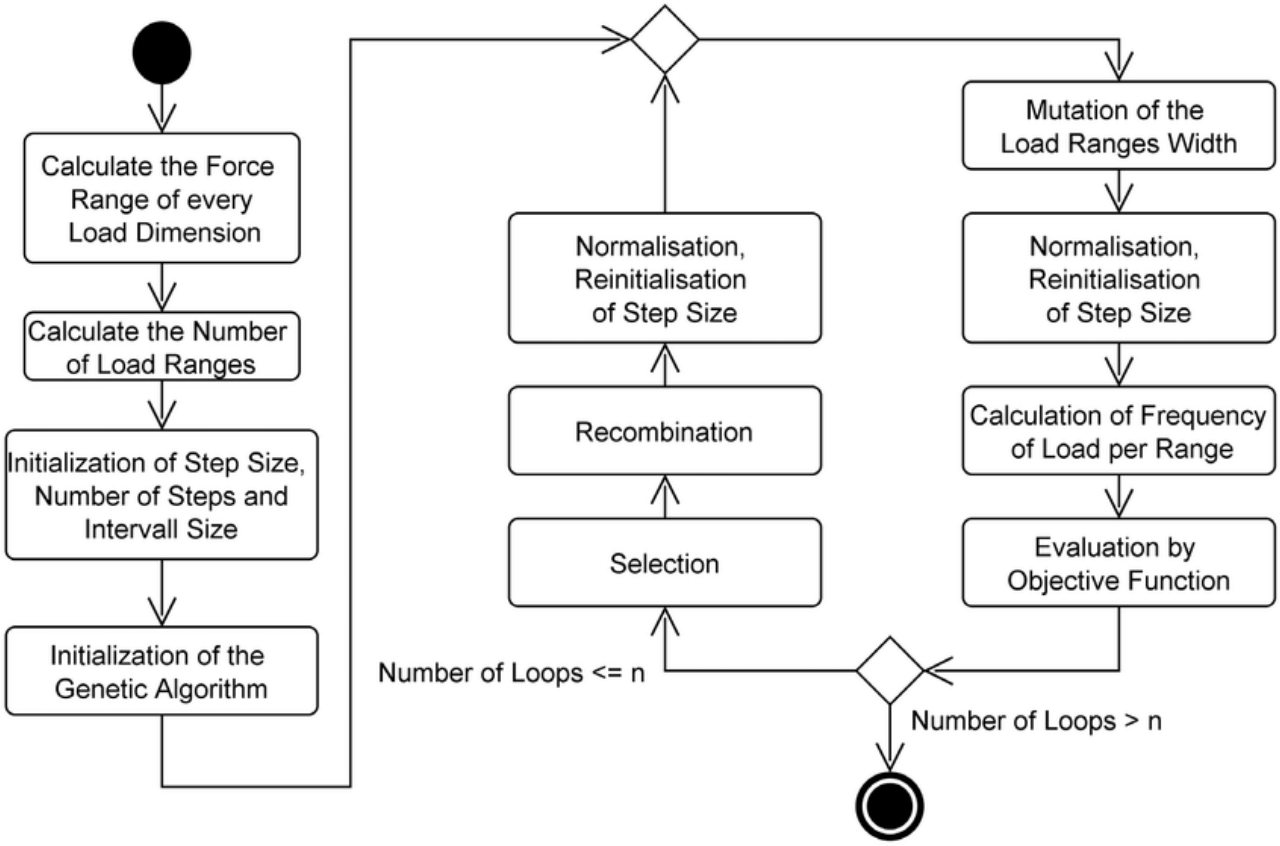

Рисунок 3.2 – Діаграма діяльності для програмного забезпечення розподілення навантаження

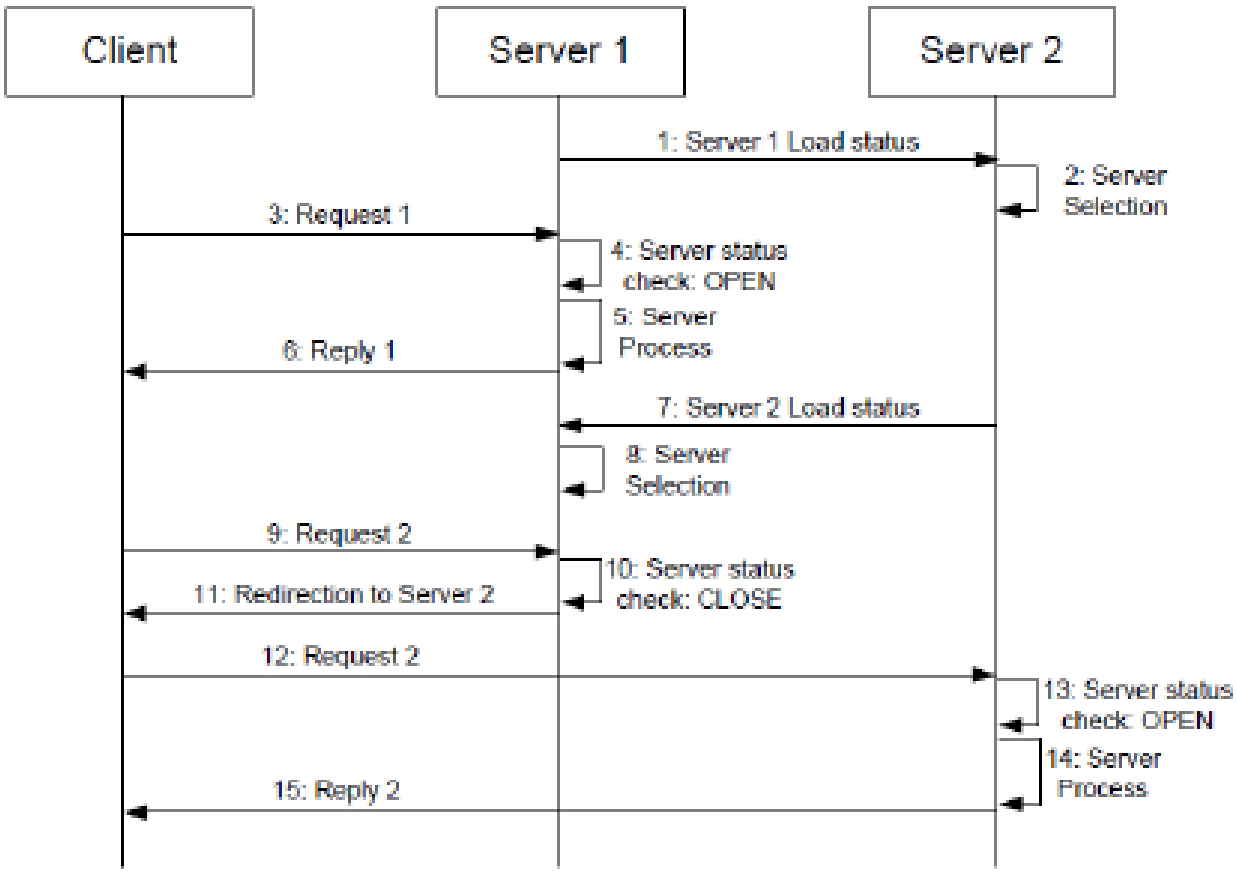

Рисунок 3.3 – Діаграма послідовності для програмного забезпечення розподілення навантаження

4 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Візуацізація генерації 8, 18 та 50 нових поколінь навелено на рисунках 4.1-4.3.

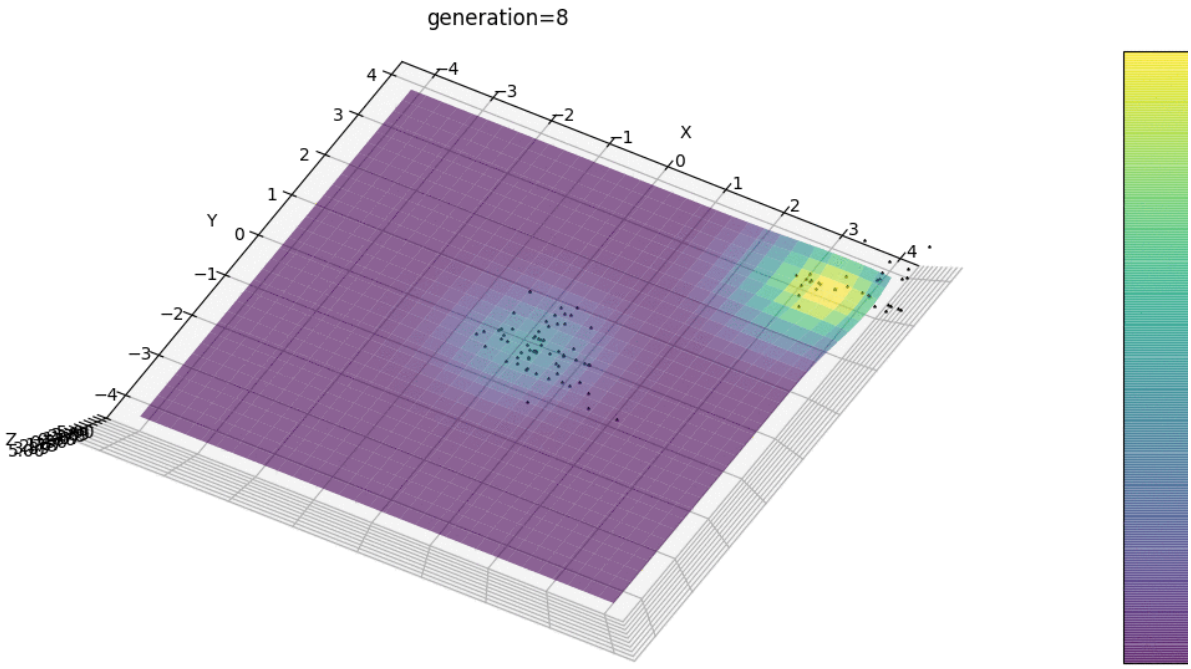

Рисунок 4.1 Генерація 8 покоління

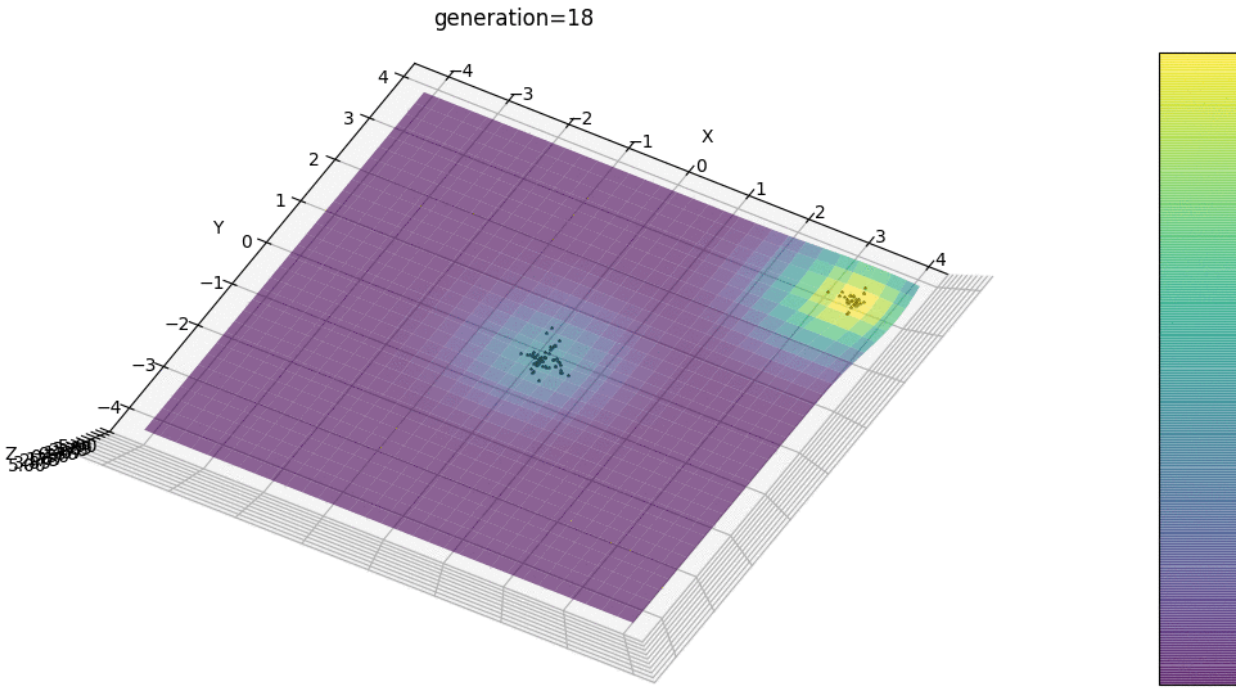

Рисунок 4.1 Генерація 18 покоління

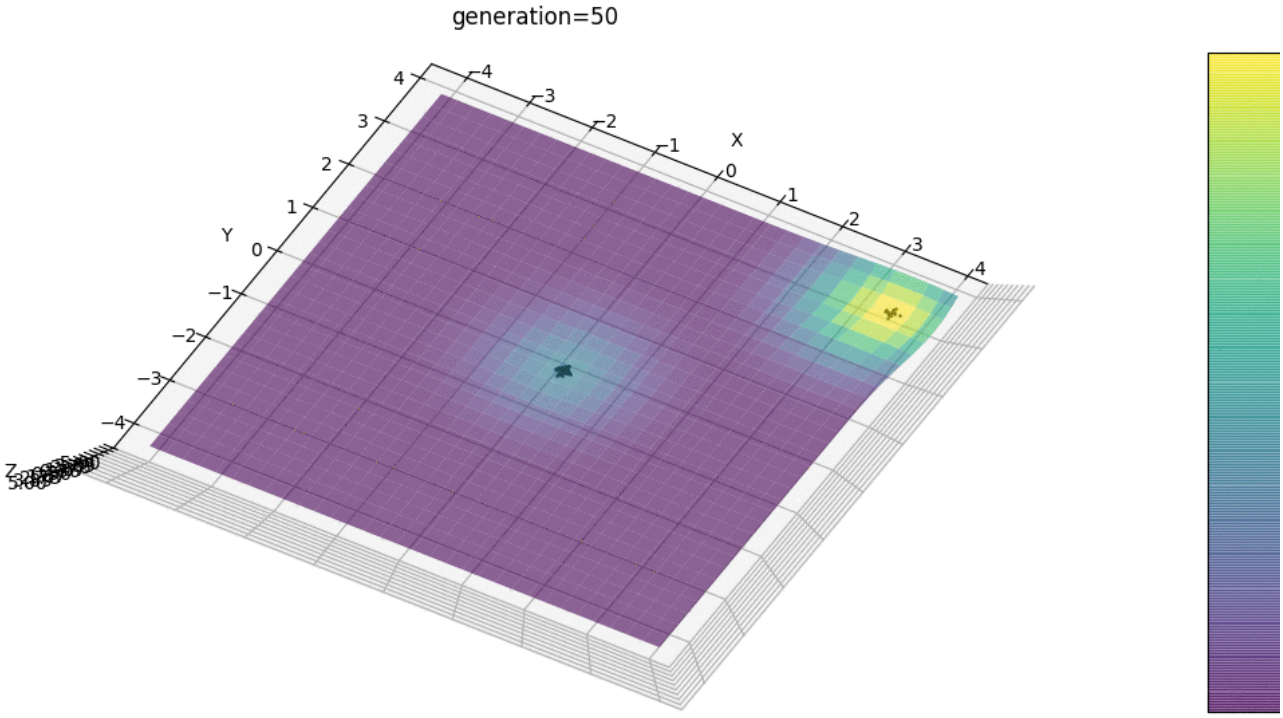

Рисунок 4.1 Генерація 50 покоління

Опис програми наведено в Додатку В.

Інструкція користувача для роботи з розробленою системою наведена в Додатку В.

Вихідний код програми наведено у Додатку Б.

5.РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Ефективність інформаційних систем визначається ступенем їх позитивного впливу на підприємство, що управляється або процес в результаті використання засобів обчислювальної техніки, програмного забезпечення, зміни інформаційних потоків та організаційної структури управління.

Створення програмного забезпечення вимагає одноразових витрат на розробку, придбання необхідних технічних засобів і поточних витрат на функціонування продукту. Економія від функціонування ПЗ визначається з урахуванням витрат на його експлуатацію. Відношення цієї економії до витрат на створення програмного забезпечення характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються за діючими на момент розрахунку оптовими цінами, тарифами і ставками заробітної плати.

5.1 Розрахунок витрат на створення й експлуатацію програмного забезпечення

Витрати на розробку системи складаються з витрат:

- на заробітну платню розробника;
- на амортизацію ЕОМ, на якій виконується розробка;
- на експлуатацію цієї ЕОМ;
- на засоби розробки;
- на матеріали і комплектуючі.

Розробка програмного забезпечення виконується програмістом, місячна заробітна плата якого складає 8000 грн.

Вартість сучасного ПК становить 15000 грн.

При вартості кіловат-години електроенергії 1.29 грн, розраховується вартість розробки програми.

Витрати на допоміжні матеріали наведені в табл. 5.1.

Таблиця 5.1 – Витрати на допоміжні матеріали

№	Пункти витрат	Сума, грн
1	Флеш-накопичувач даних	250
2	Папір для принтера	80
3	Картридж та тонер для принтера	200
	Разом	530

Вартість ПЗ знаходиться за формулою:

$$Спр = (Ззп + Зсз + Ззг + Зе) * Т + Зм \quad (5.1)$$

де Т – тривалість розробки, міс.

Ззп – основна і додаткова заробітна плата обслуговуючого персоналу, грн.

Зсз – відрахування на соціальні заходи (15% від основної і додаткової заробітної плати), грн.

Ззг – загальногосподарські витрати (10% від основної заробітної плати), грн.

Зе – витрати на електроенергію, грн.

Зм – витрати на основні і допоміжні матеріали, грн.

Зе при споживанні потужності 0.5 кВт, тривалості роботи на місяць, рівної 22*8 = 176 годин, вартості кіловат-години електроенергії 1.29 грн становить:

$$Зе = 176 * 1.29 * 0.5 = 113.52 \text{ грн}$$

Витрати на розробку ПЗ наведені в табл. 5.2.

Таблиця 5.2 – Витрати на розробку програмного забезпечення

№	Найменування витрат	Одиниця	Кількість
1	Тривалість розробки (Т)	Міс.	3
2	Основна і додаткова заробітна плата (Ззп)	Грн.	8000.00
3	Відрахування на соціальні заходи (Зсз)	Грн.	1200.00
4	Загальногосподарські витрати (Ззг)	Грн.	800.00
5	Витрати на допоміжні матеріали (Зм)	Грн.	530.00
6	Витрати на електроенергію (Зе)	Грн.	113.52

За формулою 5.1 розрахуємо вартість програми:

Спр = (8000 + 1200 + 800 + 113.52) * 3 + 530 = 30870.56 грн

Амортизаційні відрахування складають 25% балансової вартості на рік:

Аоб = 15000 * 0.25 = 3750 грн

В масштабах підприємства річні витрати на основні і допоміжні матеріали визначаються в розмірі 5% вартості основного устаткування:

Вм = 15000 * 0.05 = 750 грн

Річний обсяг робіт ПК у годинах визначається в такий спосіб:

Фм = 264.5 * ТЗ (5.2)

Тз – це середнє місячне навантаження устаткування (близько 6 годин), 264.5 – середня кількість робочих днів на рік.

Таким чином, річний обсяг роботи ПК складає:

Фм = 264.5 * 6 = 1587 годин

Витрати на електроенергію Зе при 1587 годинах роботи устаткування на рік становлять:

Зе = 1587 * 1.29 * 0.5 = 1023.62 грн

Експлуатаційні витрати на ПК на рік складуть:

Ззр = 3750 + 750 + 1023.62 = 5523.62 грн

Отже, у перший рік витрати на створення й експлуатацію ПЗ складуть:

Зр = 30870.56 + 5523.62 = 36394.18 грн

5.2 Економічна ефективність розробки і впровадження програмного забезпечення для оцінювання трудомісткості тестування Web-проектів

Основним показником економічної ефективності програмного забезпечення є зменшення трудових витрат та часу, а також ефективне розподілення трудових ресурсів на основі отриманої оцінки трудомісткості тестування.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження програмного продукту вивільняє 1.5 працівника (за експертною оцінкою фахівців підприємства).

Заробітна плата 1.5 працівника в рік складає:

З = 8000 * 12 * 1.5 = 144000 грн

Річний економічний ефект розраховується за формулою:

$$Ерік = \Delta Cn - En * k \quad (5.3)$$

де ΔCn – вивільнені кошти після впровадження продукту (144000 грн) мінус експлуатаційні витрати (36394.18 грн) = 107605.82 грн;

En – коефіцієнт ефективності (дорівнює коефіцієнту амортизації 0.25);

k – одноразові витрати на впровадження продукту (30870.56 грн).

Ерік = 144000 – 0.25 * 30870.56 = 136282.36 грн

Термін окупності ПЗ розраховується за формулою:

$$T = k / \Delta Cn = 30870.56 / 107605.82 = 0.29 \text{ року}$$

Отже, термін окупності ПЗ становить приблизно 4 місяці.

Висновки

У цьому розділі були здійснені розрахунки витрат, економічної ефективності та терміну окупності ПЗ. Виходячи з розрахунків, впровадження даного програмного забезпечення окупить себе протягом 4 місяців. Таким чином, його розробка є економічно обґрунтованою.

6 ОХОРОНА ПРАЦІ

На великих підприємствах зараз діє система заходів, яка спрямована на запобігання та попередження виникнення різних ситуацій, які можуть призвести до загрози життю і здоров’ю працівника. Дана система заходів називається охороною праці. Але потрібно більш детально охарактеризувати дану категорію.

Під охороною праці розуміється сукупність способів, засобів і дій, спрямованих на скорочення в рамках підприємств або галузей травматизму, ситуацій, які можуть надати шкоду здоров’ю простого робітника.

Законодавство по охороні праці складається з дійсних Законів України «Про охорону праці», «Про охорону здоров’я», «Про пожежну безпеку», «Про забезпечення санітарного та епідеміологічного благополуччя населення», Кодексу законів про працю України та інших нормативних актів. У випадку, коли міжнародними договорами чи угодами, у яких бере участь Україна, встановлені більш високі вимоги до охорони праці, чим ті, котрі передбачені законодавством України, застосовуються правила міжнародного договору чи угоди.

Визначення терміну «охорона праці» наведено в першій статті закону України «Про охорону праці». Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів і заходів вкладених у збереження життя, здоров'я та перемоги працездатності людини під час праці. Закон України визначає основні напрямки по реалізації конституційного права на охорону життя і здоров'я в процесі трудової діяльності, регулюють при участі відповідних державних органів відносини між власником підприємства, чи установи організації, чи органом і роботодавцем з питань зовнішнього середовища і встановлює єдиний порядок організації охорони праці в Україні [38].

6.1 Аналіз впливу шкідливих виробничих факторів в ІТ відділі компанії Basoft

Цілком безпечних і нешкідливих виробництв не існує. Головна задача охорони праці - звести до мінімальної ймовірності поразки або захворювання працюючого, а також виявлення і вивчення шкідливих факторів, їхній вплив на людину і навколишнє середовище. Виробниче середовище - це середовище, у якій людина забруднює навколишнє середовище. Небезпечним виробничим фактором називається такий виробничий фактор, вплив котрого на працюючого у визначених умовах приведе до травми або до іншому раптовому, різкому погіршенню здоров’я.

Шкідливим виробничим фактором називається такий виробничий фактор, вплив котрого на працюючого у визначених умовах призведе до захворювання або зниження працездатності. Прикладами небезпечних факторів можуть служити відкриті струмоведучі частини устаткування, що рухають деталі машин та інше. Прикладами шкідливих факторів являються шкідливі домішки в повітрі, несприятливі метеорологічні умови, шум, вібрації, недостатнє освітлення [39].

Рівень безпеки є результатом взаємодії людини і того середовища, системи безпеки, що діє на виробництві. До факторів, які впливають на виконання виробничого процесу відносять мікроклімат приміщення, шум та вібрацію, освітлення, електробезпеку, пожежну безпеку, пил та інші. Отже розглянемо їх детально.

Шумом є всякий небажаний для людини звук. Шум на виробництві завдає великої шкоди, шкідливо впливаючи на організм людини і знижуючи продуктивність праці. Шум навіть коли невеликий (при рівні 50-60 дБа), створює значне навантаження на нервову систему людини, роблячи на нього психологічний вплив. Під впливом шуму, що перевищує 85-90 дБа, у першу чергу знижується слухова чутливість на високих частотах.

Нормою виробничого шуму є рівень звуку до 85 дБ. Якщо рівень перешкод становить 20 дБ, то такий шум не заважає розбірливості мови. З підвищенням рівня перешкод до 70 дБ та вище мова стає нерозбірливою.

Джерелами шуму на ТОВ «Трістал Діджитал» є: виробниче устаткування для випікання, перемелювальні пристрої, двигуни, пневматичні та електричні інструменти, верстати, будівельна техніка тощо.

Небезпечним фактором для роботи програміста є також робота за комп’ютером. Найбільшому ризику піддаються зорова, опорно-рухова та нервово-психічна системи. Монітор випускає випромінювання декількох видів: рентгенівське, ультрафіолетове, інфрачервоне, електромагнітне. Для кожного з цих випромінювань розроблені гранично допустимі норми. Норми передбачають, що опроміненню піддається верхня частина тулуба. Згадані норми встановлені з розрахунку на кожен вид опромінення в окремо, хоча реально всі поля діють одночасно, а їх комплексний вплив досі не досліджено.

Відеоапаратура порушує рівновагу між позитивно і негативно зарядженими іонами в повітрі. Електростатичне поле дисплея притягає негативні іони, порушуючи загальний баланс атмосфери. Це також шкодить здоров'ю. Вже через годину роботи біля монітора спостерігається майже повне зникнення негативних іонів. Ось чому необхідно, щоб до робочого місця за комп'ютером проникав свіже повітря.

Освітленість робочої зони комп'ютера повинна відповідати стандартам та бути вище яскравості монітора. Відстань від очей до екрана повинна бути не менше полу метра. Висота крісла повинні бути такою щоб очі були на одному рівні з центром монітора. Фахівці говорять, що саме очі найбільш страждають при роботі з комп'ютером. Занадто відсунутий монітор призводить до перенавантаження м'язів підстави голови та шиї, через що тиск на їх роботу зростає приблизно в три рази, судини шиї здавлюються, погіршуючи кровопостачання голови. Крім того, людині, що сидить у такій незручній позі, доводиться щоразу відкидати голову назад, для фіксування уваги на ближчих

об’єктах. Це посилює прогин шийного відділу хребта. Через деякий час це може призвести до головної болі та болю у кінцівках, оскільки нерви, що відходять від спинного мозку в області шиї, простягаються до кінчиків пальців.

Малорухома та тривала сидяча поза за комп’ютером шкідлива для опорно-рухового апарату, що веде до застою крові в органах людини. Це особливо проявляється при фізіологічному положенні різних частин тіла і тривало повторюваних одноманітних рухах. Під час роботи за комп'ютером людина сидить кілька годин поспіль в незручному становищі. Це не тільки загрожує втомою і загальним знесиленням організму, а й може призвести до розвитку остеохондрозу різних ділянок хребта - шийного, грудного, попереково-крижового.

Суттєвий вплив на стан організму ІТ спеціаліста, його працездатність здійснює мікроклімат (метеорологічні умови) у відділі, під яким розуміють клімат внутрішнього середовища цих приміщень, що визначається діючою на організм людини сукупністю температури, вологи, руху повітря та теплового випромінювання нагрітих поверхонь. Несприятливі метеорологічні умови призводять до швидкої втоми працівника, збільшення захворюваності та зниження продуктивності праці.

Світло впливає не лише на функції органів зору, а й на діяльність організму в цілому. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, зростає потенційна небезпека помилкових дій і нещасних випадків. Для створення оптимальних умов зорової роботи слід враховувати не лише кількість та якість освітлення, а й кольорове оточення. Так, при світлому фарбуванні інтер’єру завдяки збільшенню кількості відбитого світла рівень освітленості підвищується на 20-40%, різкість тіней зменшується, покращується рівномірність освітлення.

При надмірній яскравості джерел світла та оточуючих предметів може відбутися засліплення працівника. Нерівномірність освітлення та неодинакова яскравість оточуючих предметів призводять до частої переадаптації очей під

час виконання роботи, і як наслідок цього - до швидкого стомлення очей. Тому поверхні, що добре освітлюються і знаходяться в полі зору, краще фарбувати в кольори середньої світлості.

При освітленні ІТ відділу відділу використовують природне освітлення, створюване світлом неба (прямими і відбитим), штучне, здійснюване електричними лампами, і сполучене, при якому у світлий час доби недостатнє по нормах, природне освітлення доповнюється штучним. [40].

Об’єктом дослідження є ТОВ «Трістал Діджитал». На території підприємства діють 2 відділи. Також у ІТ відділі не належний стан освітленості приміщення та мікроклімату. Технологічні процеси і комплекс виробничого устаткування для вироблення продукції, використовуваних на підприємстві, відповідають вимогам документів і забезпечують функціонування підприємства без порушення законодавства про охорону навколишнього середовища.

Для забезпечення здоров’я працюючих приймаються необхідні запобіжні міри захисту, що повністю або частково ізолюють доступ у зону, в якій діють шкідливі фактори, та виключають їх дію в разі проникнення людини у простір, де вони виникають [41].

Велика кількість шуму негативно впливає на розумову роботу мозку, значно знижує продуктивність роботи програміста.

До методів боротьби із шумом відносяться:

- зменшення шуму в джерелі виникнення;

- зміна напрямку випромінювання шуму;

- акустична обробка приміщень;

- установка звукоізолюючих огороджень, кожухи, екрани, кабіни;

- установка глушителів шуму;

- використання засобів індивідуального захисту (вкладиші, навушники, шоломи).

Конструктивно вони можуть бути зроблені у вигляді гратчастих, сітчастих та непрозорих перешкод із металу, деревини тощо. Віброізоляція зменшує рівні вібрації, що передаються від джерела на тіло працюючого. Вона здійснюється введенням між джерелом вібрації та працюючим проміжного пружного зв’язку

Заходи, щодо зниження негативного впливу мікроклімату:

1. впровадження раціональних технологічних процесів;

2. механізація та автоматизація виробничих процесів;

3. захист працівників різними типами екранів;

5. раціональна теплова ізоляція устаткування;

6. раціональне розміщення устаткування

7. раціональне планування та конструкторське рішення виробничих приміщень;

8. раціональні вентиляція та опалення;

Для збереження здоров’я очей, та запобіганню перевтоми робоче приміщення ІТ відділу повинно відповідати наступним вимогам:

- створювати на робочій поверхні освітленість, що відповідає характеру зорової роботи і не є нижчою за встановлені норми;

- не повинно чинити засліплюючої дії як від самих джерел освітлення, так і від інших предметів, що знаходяться в полі зору;

- забезпечити достатню рівномірність освітлення;

- не створювати на робочій поверхні тіней;

- повинно бути надійним і просторим в експлуатації, економічним та естетичним.

Робоче місце має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів:

1. Висота робочої поверхні робочого столу має регулюватися в межах 680-800 мм, а ширина і глибина – забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600-1 400мм, глибина – 800-1 000мм).

2. Робочий стіл повинен мати простір для ніг заввишки не менше ніж 600мм, завширшки не менше ніж 500мм, завглибшки (на рівні колін) не менше ніж 450мм, на рівні простягнутої ноги не менше ніж 650мм. Робочий стілець має бути підйомно-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край – заокругленим.

3. Робочі місця слід розташовувати відносно світових прорізів так, щоб природне світло падало переважно з лівого боку.

4. Монітор має розташовуватися на оптимальній відстані від очей користувача, що становить 600-700мм, але не ближче ніж за 600мм з урахуванням розміру літерно-цифрових знаків і символів. Розташування екрана монітору має забезпечувати зручність зорового спостереження у вертикальній площині під кутом +30 градусів до нормальної лінії погляду працівника.

5. Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, звернутого до працюючого. У конструкції клавіатури має передбачатися опорний пристрій (виготовлений із матеріалу з високим коефіцієнтом тертя, що перешкоджає мимовільному її зсуву), який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5-15 градусів.

На даний час, це поки що найреальніша можливість захистити людину, що працює біля комп’ютера, від професійного захворювання.

Електрика широка застосовується у всіх галузях. Тому питанню електробезпечності необхідно приділяти велику увагу. Електробезпечність - система організаційних і технічних заходів і засобів, що забезпечують захист людей від шкідливого і небезпечного впливу електричного струму, електричної дуги, та інше. Проходячи через організм, електричний струм робить термічні, електролітичні і біологічні дії. Це різноманіття дій електричного струму нерідко приводити до різних електричних травм, що умовно можна звести до двох видів: місцеві електричні травми та загальні електричні травми.

Основні поразки струмом наступні:

- випадковий дотик або наближення на небезпечну відстань до струмопровідних частин, що знаходиться під напругою;

- поява напруги на металевих корпусних частинах електроустаткування;

- поява напруги на відключених струмоведучих частинах, на яких працюють люди, унаслідок помилкового включення установки;

Основними заходами захисту від поразки струмом є: забезпечення неприступності струмоведучих частин, що знаходяться під напругою, для випадкового дотику; електричний поділ мережі; усунення небезпеки поразки з появою напруги на корпусах, кожухах і інших частинах електроустаткування, що досягається застосуванням малих напруг, використанням подвійної ізоляції, виявленням потенціалу, захисним заземленням, занулюванням, захисним відключенням і інше; застосування спеціальних електрозахисних засобів - переносних приладів і пристосувань; організація безпечної експлуатації електроустановок.

Занулюванням називається навмисне електричне з'єднання з нулем захисним провідником металевих не струмоведучих частин, що можуть виявитися під напругою. Призначення нульового захисного провідника - створення струмові короткого замикання ланцюга з малим опором, щоб цей

струм був достатнім для швидкого спрацьовування захисту, тобто швидкого відключення ушкодженої установки від мережі.

Захисне відключення - швидкодіючий захист, забезпечує автоматичне відключення при виникненні в ній небезпеки поразки струмом. Така небезпека може виникнути, зокрема, при замиканні фази на корпус електроустаткування; при зниженні опору ізоляції фаз щодо землі нижче визначеної межі; появі в мережі більш високої напруги; дотик людини до струмоведучої частини, що знаходиться під напругою. Встановлюють прилад захисного відключення ручний і автоматичний вимикач. Принцип дії - це швидке відключення від мережі установки, якщо напруга її корпусу щодо землі виявиться вище деякого гранично припустимого значення, внаслідок чого дотик до корпусу стає небезпечним.

Основним нормативним документом, що регламентує вимоги щодо пожежної безпеки є Закон України “Про пожежну безпеку”, цей закон визначає загальні правові, економічні та соціальні основи забезпечення пожежної безпеки на території України, регулює відносини державних органів, юридичних і фізичних осіб у цій галузі незалежно від виду їх діяльності та форм власності. Пожежа - це неконтрольоване горіння поза спеціальним вогнищем, що розповсюджується в часі і просторі та створює загрозу життю і здоров’ю людей, навколишньому середовищу, призводить до матеріальних збитків. Основними причинами пожежі на підприємстві є:

- небезпечне поводження з вогнем;

- незадовільний стан електротехнічних пристроїв та порушення правил їх монтажу та експлуатації;

- порушення режимів технологічних процесів;

- невиконання вимог нормативних документів з питань пожежної безпеки.

Власник підприємства, повинен:

- розробляти комплексні заходи щодо забезпечення пожежної безпеки;

- організувати навчання працівників правилам пожежної безпеки та пропаганду заходів щодо їх забезпечення;

- повинен створити засоби протипожежної безпеки.

Система протипожежного захисту - це сукупність організаційних заходів, а також технічних засобів, спрямованих на запобігання впливу на людей небезпечних факторів пожежі та обмеження матеріальних збитків від неї. На підприємстві у кожному цеху є два вогнегасника.

У результаті проведеного дослідження можна зробити наступні висновки. Під час розумової або фізичної праці людина сприймає комплекс шкідливих виробничих чинників, які можуть позитивно або негативно відзначитись на її здоров'ї та продуктивності праці. Рівень цих факторів залежить від виду робочої діяльності, але можна сказати, що навіть робота в офісі може нашкодити здоров'ю працівника.

На ТОВ «Трістал Діджитал» шкідливих чинників дуже багато, та їх дія безпосередньо негативно впливає на організм людини в цілому, та на окремі його органи. Постійний вплив цих факторів може призвести до професійних або хронічних захворювань.

Працівникам на потенційно небезпечному виробництві або з небезпечним обладнанням необхідно пам'ятати про засоби індивідуального захисту, правила поводження зі спеціальним обладнанням, адже від цього напряму залежить особисте здоров'я працівника.

Керівництву підприємства необхідно забезпечувати працівників та робітників необхідними засобами індивідуального захисту, вчасно слідкувати за станом робочих місць, який відповідає інструкції з техніки безпеки та не шкодить здоров'ю людини.

6.2 Заходи по створенню безпечних та здорових умов праці

Відділ охорони праці компанії Basoft створено відповідно до Закону України „Про охорону праці” з метою організації та виконання ряду правових, організаційно-технічних, санітарно-гігієнічних, соціально-економічних і лікувально-профілактичних заходів, спрямованих на запобігання нещасним випадкам, професійних захворювань та аварій в процесі праці.

Робота відділу охорони праці підприємства повинна здійснюватися відповідно до плану роботи і графіків обстежень, затверджених працедавцем. Робочі місця працівників відділу охорони праці розміщуються в окремому приміщенні, забезпечуються належною оргтехнікою, технічними засобами зв’язку і створені умови для прийому відвідувачів. Відділ охорони праці взаємодіє з іншими структурними підрозділами, службами, фахівцями підприємства і представниками профспілки.

Відділ охорони праці виконує наступні функції [39]:

1. Здійснює контроль за дотриманням в підрозділах підприємства вимог чинного законодавства, інструкцій, правил та норм з охорони праці, техніки безпеки, виробничої санітарії, протипожежному захисту.

2. Контролює виконання розділу „Охорона праці” колективного договору та актів з охорони праці, що діють в межах підприємства.

3. Організовує проведення профілактичних заходів, спрямованих на усунення шкідливих та небезпечних виробничих чинників, запобігання нещасним випадкам на виробництві, професійних захворювань та інших випадків загрози життю або здоров’ю працівників.

4. Проводить ввідний інструктаж з охорони праці, техніки безпеки з працівниками, що знов поступають на роботу, здійснює перевірки своєчасності проведення та якості інструктажу на робочих місцях.

5. Забезпечує ведення обліку і проведення, аналізу причин виробничого травматизму, професійних захворювань, аварій, заподіяної ними шкоди.

6. Інформує працівників про основні вимоги законів, інших нормативно-правових актів і актів по охороні праці, що діють в межах підприємства.

7. Розглядає:

- питання про підтвердження наявності небезпечної виробничої ситуації, яка стала причиною відмови працівника від виконання дорученої роботи, відповідно до законодавства (у разі потреби);

- листи, заяви, скарги працівників підприємства, що стосуються питань охорони праці.

8. Організовує:

- забезпечення підрозділів нормативно-правовими актами і актами з охорони праці, що діють в межах даного підприємства, допомогою, учбовими матеріалами з цих питань;

- роботу кабінету охорони праці, підготовку інформаційних стендів, куточків охорони праці;

- наради, семінари, конкурси з питань охорони праці, пропаганду з питань охорони праці з використанням інформаційних засобів.

9. Бере участь в:

- розслідуванні нещасних випадків, професійних захворювань і аварій на виробництві.

- складанні санітарно-гігієнічної характеристики робочих місць для тих працівників, які проходять обстеження щодо наявності профзахворювань.

10. Здійснює контроль за:

- виконанням заходів, передбачених програмами, планами поліпшення стану безпеки, гігієни праці та виробничого середовища, колективним

договором і заходами, направленими на усунення причин нещасних випадків і професійних захворювань;

- проведенням ідентифікації, декларування безпеки об’єктів підвищеної небезпеки;

- використанням цільових засобів, виділених для виконання комплексних мерів для досягнення встановлених нормативів і підвищення існуючого рівня охорони праці;

- виконанням розпоряджень посадових осіб органів державного нагляду за охороною праці та представлень страхових експертів з охорони праці;

- проведенням попередніх (при прийомі на роботу) та періодичних (протягом трудової діяльності) медичних оглядів працівників, зайнятих на важких роботах, роботах з шкідливими або небезпечними умовами праці..

6.3. Розрахунок штучного освітлення приміщення ІТ відділу компанії Basoft

Проведемо розрахунок штучного освітлення приміщення ІТ відділу компанії Basoft за методом коефіцієнта використання світлового потоку [52, с. 39-56]. Довжина приміщення (A) = 5 м, ширина (B) = 4 м, висота (H) = 3 м, висота робочої поверхні (h_p)= 0,8 м, для освітлення приймаємо світильник типу УПД, мінімальна освітлюваність лампи розжарювання за нормами E_{min}=100лк, коефіцієнт відображення стелі $\rho_п$ =70%, стін $\rho_с$ = 50%, робочої поверхні $\rho_р$=30%. Напруга мережі 200 Вт.

Розрахуємо відстань від стелу до робочої поверхні (H_0):

$$H_0 = H - h_p, \tag{6.1}$$

де H - висота приміщення, м;

h_p - висота робочої поверхні, м.

$$H_0 = 3 - 0{,}8 = 2{,}2 \text{ м.}$$

Визначимо відстань від стелі до світильника (h_c):

$$h_c = 0{,}2 \times H_0, \tag{6.2}$$

$$h_c = 0{,}2 \times 2{,}2 = 0{,}44 \text{ м.}$$

Висота підвішування світильника над освітлюваною поверхнею (h):

$$h = H_0 - h_c, \tag{6.3}$$

$$h - 2{,}2 - 0{,}44 - 1{,}76 \text{ м.}$$

Висота підвішування світильника над підлогою (H_n):

$$H_n = h + h_p, \tag{6.4}$$

$$H_n = 1{,}76 + 0{,}8 = 2{,}56$$

Для того, щоб досягти найбільшої рівномірності освітлення приймаємо відношення L/h = 1,6

Тоді відстань між центрами світильників

$$L = 1{,}6 \times h = 1{,}6 \times 1{,}76 = 2{,}82 \text{ м.}$$

Визначимо необхідну кількість світильників [52, с. 55]:

$$N = \frac{S}{L^2}, \tag{6.5}$$

$N = \frac{5 \times 4}{2{,}82^2} = \frac{20}{7{,}93} = 2{,}52$ приймаємо 3 шт.

Знаходимо індекс приміщення:

$$\text{i} = \frac{\text{A} \times \text{B}}{[\text{h} \times (\text{A} + \text{B})]} = \frac{5 \times 4}{1{,}76 \times (5 + 4)} = \frac{20}{35{,}2} = 0{,}57.$$

При i = 0,57, коефіцієнтах відображення стелі $\rho_п$ =70%, стін $\rho_с$ = 50%, робочої поверхні $\rho_р$=30% для світильника УПД коефіцієнт використання світлового потоку η = 0,28.

Світловий потік однієї лампи, лм:

$$Ф = \frac{E_{min} \times S \times Z \times K_з}{N \times \eta \times \gamma}, \quad (6.6)$$

де E_{min} - рівень мінімальноїосвітленості за нормами, лк;

S - освітлювана площа приміщення, $м^2$;

$К_з$ - коефіцієнт запасу;

Z - коефіцієнт мінімальної освітленості;

N - число світильників;

η - коефіцієнт використання світлового потоку ламп, встановлених у світильнику.

Коефіцієнт запасу $К_з$ враховує експлуатаційне зниження освітленості, в порівнянні із запроектованою, внаслідок забруднення, а також зменшення світлового потоку ламп у процесі їх експлуатації. Для нашого приміщення коефіцієнт мінімальної освітленості ($К_з$) = 1,3.

Коефіцієнт мінімальної освітленості Z дорівнює відношенню середньої освітленості до мінімальної. Він враховує нерівномірність освітленості і залежить в основному від відношення відстаней між світильниками і від їх типів. Значення цього коефіцієнту для ламп розжарювання приймають Z =1,15.

$$Ф = \frac{100 \times 20 \times 1{,}15 \times 1{,}3}{3 \times 0{,}28} = \frac{2990}{0{,}84} = 3559{,}5.$$

За знайденим світловим потоком вибираємо лампу потужністю 200 Вт, що має світловий потік 3 200 лм (Г125-135-200) найбільш близький до розрахункового [52, с. 24].

При цьому фактична освітленість (лк) дорівнює:

$$E_{\phi} = E_{\min} \times \frac{\Phi_{л}}{\Phi_{р}}, \quad (6.7)$$

де $\Phi_{л}$ - світловий потік обраної лампи, лм;

$\Phi_{р}$ - світловий потік лампи, отриманої розрахунком,лм.

$$E_{\phi} = 100 \times \frac{3200}{3559} = 89,9.$$

Загальна потужність освітлювальної установки $P_з$,Вт, визначається за формулою:

$$P_з = P_л \times N, \quad (6.8)$$

де $P_з$ - потужність обраної лампи,Вт.

$$P_з = 200 \times 3 = 600 \text{ Вт} = 0,6 \text{ кВт}.$$

6.4 Висновки до розділу 6

У даному розділі визначені цілі, завдання та функції відповідального з охорони праці компанії Basoft, описані шкідливі та небезпечні фактори приміщення ІТ відділу підприємства, а також розглянуті заходи щодо мінімізації їхнього впливу на працівників. Також був проведений розрахунок штучного освітлення, в результаті якого було визначено, що для освітлення відділу кадрів необхідно встановити 3 світильники типу УПД потужністю 200 Вт і світловим потоком 3 200 лм. Загальна потужність освітлювальної установки становитиме при цьому 600 Вт.

7. Охорона навколишнього середовища

Вступ

Під навколишнім середовищем розуміють цілісну систему взаємопов’язаних природних і антропогенних об’єктів і явищ, під впливом і при безпосередньому використанні яких відбувається праця, побутова діяльність, відпочинок людей. Поняття «навколишнє середовище» включає соціальні, природні і штучно створені фізичні, хімічні та біологічні фактори, тобто все те, що впливає на життя і діяльність людини. Складовою частиною навколишнього середовища є природне середовище. Перед сучасним суспільством стоїть завдання не тільки зберегти природу, а й запобігти негативним наслідкам господарської діяльності людини в майбутньому.

Економічна діяльність у всіх її проявах здійснює забруднення навколишнього середовища. У процесі цієї діяльності забруднюються і стають дефіцитними ресурси повітря, води, територій, що здавалися нескінченними. Нині рівень забруднення досяг загрозливих розмірів, набувши по суті кризового характеру.

Цей процес посилюється розвитком виробничих сил і збільшенням маси речовин, що залучаються в господарський обіг. Через це в навколишнє середовище надходить все більше й більше різноманітних речовин, які йому чужі, а часом токсичні. Значна частина з них не включається в природний кругообіг, накопичується в біосфері і зумовлює небажані екологічні наслідки. Відомо, що екологія – це наука взаємовідносин між живими організмами і сферою їх перебування, тому наслідки промислової і господарської діяльності людства можуть завдати непоправні збитки біосфері і велику шкоду людині.

Забруднюючі речовини, що потрапляють в природне середовище здатні переміщуватись на досить великі відстані, а закономірність цих процесів вивчена ще недостатньо. Ці речовини мігрують у великих кількостях в контурах окремих складових біосфери. Так в атмосфері вони переносяться повітряними течіями. Ступінь їх розсіювання формується швидкістю і напрямом переміщення повітряних мас і залежить від метеорологічних умов. потрапивши у воду, забруднюючі речовини окиснюються мікроорганізмами або адсорбуються частинками речовин, які є у воді [28].

Охорона навколишнього середовища являє собою форму відносин між суспільством і природою. Вона здійснюється різними засобами: економічними, правовими, науково-технічними, санітарно-гігієнічними, біологічними та іншими.

В загальному випадку проблема охорони навколишнього середовища зводиться до вирішення двох завдань:

1) організації раціонального природокористування;

2) забезпечення чистоти природних (екологічних) систем.

При здійсненні різних видів економічної діяльності суб'єкти господарювання використовують різноманітні природні ресурси: землю, воду, корисні копалини тощо. Проте ресурси ці обмежені. Обмеженість природних ресурсів була і залишається головною і дуже жорсткою умовою, що накладається на розвиток економіки і відповідно зростання суспільного добробуту.

Наслідком обмеженості природних ресурсів є конкуренція за їх застосування, тобто суперництво між альтернативними цілями використання

ресурсів. Адже майже всі ресурси можуть використовуватися для задоволення найрізноманітніших потреб.

Раціональне природокористування означає розробку та здійснення концепції і конкретних заходів щодо раціонального використання і відтворення природних ресурсів, гармонічну взаємодію суспільства і природи, людини і навколишнього природного середовища.

Завдання організації раціонального природокористування вирішується шляхом:

1) оптимального розподілу ресурсів між різними господарськими цілями;

2) використання технологій, що зберігають ресурси;

3) проведення заходів щодо поповнення природних ресурсів.

Іншим, не менш важливим, завданням охорони навколишнього середовища є забезпечення чистоти природних екологічних систем, тобто водного середовища, повітряного басейну, ґрунтових покровів тощо, з тим, щоб забезпечити населення екологічно чистими продуктами харчування, водою, повітрям і, в остаточному підсумку, зберегти високий рівень здоров'я населення та його активного довголіття.

7.1 Забруднення навколишнього середовища підприємством «GFL»

Підприємство «GFL» – це офісна компанія, яка займається наданням послуг з розробки та супроводження програмного забезпечення. Хоча компанія не займається промисловим виробництвом з викидом шкідливих речовин у атмосферу, воду та ґрунт, це не виключає існування джерел забруднення довкілля. До таких джерел, насамперед, належать:

- дизельна електростанція;

- твердопаливні котли;

- побутове сміття;

- електромагнітне випромінювання.

Одночасно в офісі компанії можуть працювати до 300 комп’ютерів, стаціонарних телефонів та велика кількість серверної техніки, яка обслуговує роботу цього обладнання, аварійне чи планове відключення електроенергії призводить до значних перешкод у роботі персоналу. Для того, щоб уникнути таких ситуацій, на території компанії знаходиться дизельна електростанціє. Вона працює як в аварійному, так і в резервному режимі.

Результатом роботи дизельної електростанції є продукти згоряння дизельного палива та шумове забруднення.

Твердопаливні коти використовуються в холодну пору року в системі опалення офісних приміщень. Для опалення використовуються деревні пелети. Хоча таке джерело тепла має менший відсоток шкідливих викидів в атмосферу, ніж, наприклад, вугілля чи мазут, цей вид опалення не можна вважати екологічно чистим. Порівняльні характеристики продуктів згоряння палива наведені в табл. 7.1.

Таблиця 7.1 – Рівні викидів забруднюючих речовин в атмосферу при спалюванні різних видів палива

Вид палива	Викиди забруднюючих речовин в атмосферне повітря без систем очищення, тонн на 1 тис. тонн нат. палива				
	CO2	NO2	SO2	Тверді частинки (пил неорг.)	РАЗОМ
Природний газ	1,18	3,52	0,00	0,00	4,70
Древні брикети, пелети	**4,68**	**9,31**	**0,28**	**4,11**	**17,69**
Деревина дров'яна	4,9	9,4	0,3	4,3	18,9
Тирса деревна	5,0	9,6	0,5	5,0	20,0

Деревні відходи, обрізки	5,2	9,9	0,4	5,2	20,7
Швидкозростаюча деревина	4,8	9,5	0,0	8,4	22,7
Тріска, сучки, кора	5,6	11,4	0,8	13,4	31,3
Мазут	5,20	5,20	35,30	0,30	45,90
Брикет торф'яний	8,04	26,81	3,00	13,02	50,87
Кам'яне вугілля	9,58	63,56	9,20	65,32	147,66

Побутові відходи – це залишки речовин і предметів, які утворюються в результаті побутової та господарської діяльності людини і які не можуть бути використані на місці утворення, а їх накопичення і зберігання порушують санітарний стан навколишнього середовища.

Всі побутові відходи, які утворюються в процесі функціонування компанії, можна розділити на рідкі та тверді. До рідких побутових відходів належать нечистоти з вигребів туалетів, помиї (від миття посуду, підлоги, тощо) і стічні води (побутові, зливові). До твердих побутових відходів можна віднести сміття (побутові відходи) та кухонні відходи. Зокрема, твердими відходами є папір, картон, скло, пластмаса, продукти харчування.

Генератором електромагнітного забруднення є, в першу чергу, велика кількість комп’ютерної техніки на території офісу. Більшість із них працює 8 годин на добу, а деякі не вимикаються цілодобово. Випромінювачами в даному випадку є і процесор, і монітор. Випромінювання останнього значно вищі, особливо його бічні і задні стінки, адже вони не мають спеціального захисного покриття, як у лицьовій частині монітора. Крім того, в офісі є дві кухні, кожна з яких обладнана 5 мікрохвильовими печами.

Електромагнітне випромінювання має несприятливий вплив на організм людини. Його наслідками можуть бути головний біль, порушення сну, перевтома і, навіть, у деяких випадках стенокардія.

7.2 Розробка заходів щодо зменшення забруднення

Для того, щоб зменшити рівень забруднення навколишнього середовища, варто вжити певних заходів, які мінімізують цей вплив, якщо його неможливо усунути цілком. Серед таких заходів:

1. Очищення димових газів у мокрих золоуловлювачах.

Електрофільтри на електростанціях застосовуються для досягнення найбільш глибокого очищення димових газів в основному на великих енергоблоках потужністю 300 МВт та більше. Мокрі золоуловлювачі, які працюють при маленьких питомих витратах води та невеликих перепадах тиску, встановлюються за котлоагрегатами середньої паропродуктивності. У мокрих золоуловлювачах уловлювання часток золи на плівці води, яка стікає по його стінках, здійснюється за рахунок відцентрової сили, яка діє на частки. Ефективність апарата не перевищує 90%.

2. Використання природного газу для опалення офісних приміщень замість деревних пелетів.

Згідно з показниками, наведеними в табл. 7.1, рівень шкідливих викидів від згоряння природного газу є найнижчим серед перерахованих видів палива. Проте зміну джерела енергії слід проводити, враховуючи економічну ефективність, оскільки вартість цих видів палива не є однаковою.

3. Раціональне позбавлення від побутових відходів.

Деякі побутові відходи можна безпечно спалювати для отримання енергії. Вторинну сировину (макулатуру, скло, пластмаси) можна відправляти на переробку, а не вивозити на сміттєзвалища. Крім того, зменшити кількість побутових відходів допоможе повторне їх використання (наприклад, скляних

чи пластикових пляшок та контейнерів для їжї), скорочення споживання та використання меншої кількості упаковки.

4. Зменшити рівень електромагнітного забруднення та захиститися від його надмірного впливу.

Зрозуміло, що неможливо повністю обійтися без електроприладів та комп’ютерів, проте можна дещо зменшити їх негативний вплив. Для цього слід вимикати з електромережі комп’ютери, телефони та обладнання, з якими ніхто не працює. Також варто час від часу виходити на прогулянки і робити перерви в роботі з комп'ютером.

Що стосується мікрохвильових печей, то їх потужність може змінюватись, тому час від часу треба звертатися до майстра, щоб контролювати рівень випромінювання.

ВИСНОВКИ

В результаті виконання даної роботи було підвищено швидкість обробки запитів задяки розподіленню навантажень на складові інформаційно-вимірювальної системи за допомогою використання генетичного алгоритму.

У вирішенні поставлених завдань використано методи теорії прийняття рішень, теорія ймовірності, математичної статистики,оптимізаційні методи.

Розроблено модель розподілення навантажень інформаційно-вимірювальних систем за допомогою використання генетичного алгоритму, яка в порівнянні з моделлю, що базується на моделі замкненої мережі систем масового обслуговування має більший відсоток достовірних прогнозованих результатів, менше значення середньої величини відносної похибки.

Апробація результатів роботи. Робота пройшла апробацію на конференції Комп’ютерна інженерія і кібербезпека : досягнення та інновації : матеріали II Всеукр. наук.-практ. конф. здобувачів вищої освіти й молодих учених, м. Кропивницький, 25–27 листоп. 2020 р.

Список використаної літератури

1. Кини Р.Л., Райфа Х. Принятие решений при многих критериях: предпочтения и замещения. — М: Радио и связь, 1981. — 560 с.
2. Болотова Л. С. Системы искусственного интеллекта: модели и технологии, основанные на знаниях / Л. С. Болотова – Москва: «Финансы и Статистика», 2012. – 663
3. Терано Т. Прикладные нечеткие системы. : Пер. с яп. / Терано Т., Асаі К., Сугено М., Чернишова Ю. М.: «Мир» - Москва, 1993. 184 с.: іл.
4. Глибовець М.М., Олецький О.В. Штучний інтелект: Підручник для студентів, що навчаються за спец. "Комп’ютерні науки" та "Прикладна математика". – К.: Вид. дім "КМ Академія", 2002. – 366 с.
5. Bremermann HJ, Roghson J., Salaff S. Global properties of evolution processes. Natural automata and useful simulations. / Bremermann HJ, Roghson J., Salaff S. – London: Macmillan, 1966. – pp 3-42.
6. Комп'ютерні системи штучного інтелекту. Методичні вказівки до виконання лабораторних робіт студентами денної та заочної форми навчання спеціальностей 123 "Комп’ютерна інженерія", 122 "Комп’ютерні науки та інформаційні технології" / Укл.: Є.В. Мелешко – Кіровоград: КНТУ, 2016. – 61 с.
7. Інформаційні та комунікаційні середовища. **Погорелый** С.**Д**., **Билоус Р**.**В**. **Генетический алгоритм балансировки нагрузки** в **сети**. - С. 84-87.
8. Тененев В. А., Паклин Н. Б. Гибридный генетический алгоритм с дополнительным обучением лидера // Интеллектуальные системы в производстве. 2003. № 2. С. 181–206.
9. Дмитриев С. В., Тененев В. А. Применение прямых методов оптимизации в гибридном генетическом алгоритме // Интеллектуальные системы в производстве. 2005. - № 2. - С. 11–22.
10. Лесин В. В., Лисовец Ю. П. Основы методов оптимизации. М.: Изд-во МАИ, 1995. - 344 с.
11. Виробниче освітлення [Електронний ресурс]. – Режим доступу: http://posibnyky.vntu.edu.ua/bjd/45.htm – Заг. з екрану

12. Гігієна і фізіологія праці [Електронний ресурс]. – Режим доступу: http://intranet.tdmu.edu.ua/data/kafedra/internal/hihiena/lectures_stud/uk/med/lik/ntn/Гігієна та екологія/3/06.Гігієна і фізіологія праці.htm – Заг. з екрану
13. Анализ условий труда на рабочем месте пользователя ПЭВМ [Електронний ресурс]. – Режим доступу: http://www.techverious.ru/varwems-721-1.html – Заг. з екрану
14. Джерела і фактори забруднення навколишнього середовища [Електронний ресурс]. – Режим доступу: http://manyava.ucoz.ua/publ/dzherela_i_faktori_zabrudnennja_navkolishnogo_seredovishha/14-1-0-238 – Заг. з екрану

Ключові слова: МОДЕЛЬ РОЗПОДІЛЕННЯ НАВАНТАЖЕНЬ, ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНА СИСТЕМА, ГЕНЕТИЧНИЙ АЛГОРИТМ.

ДОДАТОК А - ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Назва програми: «GA».

Область застосування: інформаційно-вимірювальні системи.

Програма призначена для оптимізації навантаження складових інформаційно-вимірювальних систем.

1 Підстави для розробки

Підставою для розробки програмного продукту є завдання на магістерську роботу «Розробка моделі розподілення навантажень інформаційно-вимірювальних систем за допомогою використання генетичного алгоритму». За наказом 107уч від 26.10.2020

2 Призначення програми:

2.1 Функціональне призначення

Даний програмний продукт призначений для для оптимізації навантаження складових інформаційно-вимірювальних систем.

2.2 Експлуатаційне призначення

Програма може використовуватися для систем де розподілення навантаженя є актуальним.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу функцій, що виконуються

Програма повинна забезпечувати можливість виконування наведених нижче функцій:

— передбачуваність: роззподілення навантаження повинно проводитися згідно згідно з наданими критеріями відбору

— рівномірна завантаження ресурсів системи;

— масштабуємість: алгоритм повинен зберігати працездатність при збільшенні навантаженняправедливість: потрібно гарантувати, щоб на обробку кожного запиту виділялися системні ресурси і не допустити виникнення ситуацій, коли один запит обробляється, а всі інші чекають своєї черги;

— ефективність: всі сервери, які обробляють запити, повинні бути зайняті на 100%; бажано не допускати ситуації, коли один з серверів простоює в очікуванні запитів на обробку (відразу ж обмовимося, що в реальній практиці ця мета досягається далеко не завжди);

— скорочення часу виконання запиту: потрібно забезпечити мінімальний час між початком обробки запиту (або його постановкою в чергу на обробку) і його завершення;

— скорочення часу відгуку: потрібно мінімізувати час відповіді на запит користувача.

3.1.2 Вимоги до організації вхідних та вихідних даних

На сервері де проводиться запуск повинний бути встановлений пакет geneOptimizer в env

При запуску програми в корневому каталозі генетичного оптимізатора / повинні бути папки generations and results

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійності функціонування програми

Надійне (стійке) функціонування програми має бути забезпечено надійним (стійким) функціонуванням операційної системи.

3.2.2 Відмова виконання програмного продукту через некоректні дії користувача.

Відмова програми можлива внаслідок некоректних дій користувача при взаємодії з операційною системою. Щоб уникнути виникнення відмов програми за вказаною вище причини необхідно забезпечити стабільне функціонування операційної системи.

3.3 Умови експлуатації

Кліматичні умови експлуатації програмного продукту відповідають умовам експлуатації апаратної частини персонального комп'ютера.

3.5 Вимоги до інформаційної та програмної сумісності

Pycharm версії 2020 і вище

3.6 Вимоги до захисту інформації та програм

Вимоги до захисту інформації та програм відсутні.

3.7 Вимоги до маркування та упаковки

3.7.1 Вимоги до маркування:

Вимоги до маркування відсутні.

3.7.2 Вимоги до упаковки:

Вимоги до упаковки відсутні.

3.8 Вимоги до транспортування та зберігання

Вимоги до транспортування програми відсутні.

4 Вимоги до програмної документації

Склад програмної документації:

- технічне завдання;
- текст програми;
- опис програми;
- програма і методика випробувань;
- інструкція користувача.

5 Техніко-економічні показники

Економічна ефективність впровадження програмного забезпечення розрахована у розділі 5.

6 Стадії та етапи розробки програмного забезпечення

Стадії та етапи розробки програми для для оптимізації навантаження складових інформаційно-вимірювальних систем зазначені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадії розробки	Етапи робіт	Вміст робіт	Контрольні точки
Технічне завдання	Обумовлення необхідності розробки	Постановка задачі. Збір початкових матеріалів. Вибір і обумовлення критеріїв ефективності та якості розроблюваного ПЗ.	З 01.09.2020 до 05.09.2020
	Розробка та затвердження технічного завдання	Визначення вимог до ПЗ. Визначення вимог до ПЗ. Обговорення та затвердження технічного завдання	З 6.09.2020 до 17.10.2020
Ескізний проект	Розробка ескізного проекту	Попередня розробка структури вхідних та вихідних даних, уточнення методів рішення завдань, загальний опис алгоритму	З 18.10.2020 до 28.10.2020
	Затвердження ескізного проекту	Розробка пояснювальної записки. Узгодження і затвердження ескізного проекту	З 29.10.2020 до 8.11.2020
Технічний проект	Розробка технічного проекту	Уточнення структури вхідних та вихідних даних, розробка алгоритму рішення задачі, розробка структури ПЗ	З 9.11.2020 до 12.11.2020
	Затвердження технічного проекту	Розробка пояснювальної записки. Узгодження і затвердження технічного проекту	З 13.11.2020 до 14.11.2020
Робочий проект	Розробка ПЗ	Програмування та налагодження програми	З 15.11.2020 до 25.11.2020
	Розробка програмної документації	Розробка програмної документації відповідно до вимог ГОСТ 19.101-77	З 16.01.2020 до 25.11.2020
	Випробування програми	Розробка, узгодження та затвердження програми і методики випробувань, коригування програми і програмної документації за результатами випробувань	З 26.11.2020 до 1.12.2020

7 Порядок контролю та приймання

Контроль здійснюється замовником на кожній стадії розробки ПЗ. Приймання здійснюється замовником за програмою та методикою випробувань. Випробування проводиться в зазначений термін.

Хід проведення приймально-здавальних випробувань документують за допомогою протоколу проведення випробувань. На підставі протоколу проведення випробувань виконувач сумісно з замовником підписують акт приймання-здачі програми в експлуатацію.

ДОДАТОК Б ТЕКСТ ПРОГРАМИ

geneticOptimizerGUI.py

```
class gaGUI():

def __init__(self):

self.root = Tk()

self.root.title("Settings")

self.root.geometry('550x300+100+100')

self.root.configure(bg='white')

# photo = PhotoImage(file = "logo.png")

# w = Label(self.root, image=photo)

# w.grid()

self.f = StringVar()

self.chromosomes_number = IntVar()

self.generations_number = IntVar()

self.optimizer = StringVar()

self.mutation = BooleanVar()

self.mutation_range = IntVar()

self.statistics = BooleanVar()

self.save = BooleanVar()

self.plot = BooleanVar()


self.chromosomes_number.set(4)

self.generations_number.set(10)

self.optimizer.set('min')

self.mutation.set(1)
```

```
self.mutation_range.set(2)

self.statistics.set(1)

self.plot.set(1)

self.save.set(0)


self.chk1 = Checkbutton(text="Mutation", variable=self.mutation, onvalue=1, offvalue=0)

self.chk2 = Checkbutton(text="Show statistics", variable=self.statistics, onvalue=1,
offvalue=0)

self.chk3 = Checkbutton(text="Save all files", variable=self.save, onvalue=1, offvalue=0)

self.chk4 = Checkbutton(text="Show plot", variable=self.plot, onvalue=1, offvalue=0)


self.f_label = Label(text="Input f(x, y) in python format:", font='arial 13')

self.chromosomes_number_label = Label(text="Input individs number (multiple of 4):",
font='arial 13')

self.generations_number_label = Label(text="Input generations number:", font='arial 13')

self.mutation_range_label = Label(text="Input mutation range:", font='arial 13')

self.optimizer_label = Label(text="Input optimizer (min or max):", font='arial 13')


self.f_label.grid(row=0, column=0, sticky="w")

self.chromosomes_number_label.grid(row=1, column=0, sticky="w")

self.generations_number_label.grid(row=2, column=0, sticky="w")

self.optimizer_label.grid(row=3, column=0, sticky="w")

self.mutation_range_label.grid(row=5, column=0, sticky="w")


self.f_entry = Entry(textvariable=self.f)

self.chromosomes_number_entry = Entry(textvariable=self.chromosomes_number)

self.generations_number_entry = Entry(textvariable=self.generations_number)

self.mutation_range_entry = Entry(textvariable=self.mutation_range)

self.optimizer_entry = Entry(textvariable=self.optimizer)
```

```
self.f_entry.grid(row=0,column=1, padx=5, pady=5)

self.chromosomes_number_entry.grid(row=1,column=1, padx=5, pady=5)

self.generations_number_entry.grid(row=2,column=1, padx=5, pady=5)

self.optimizer_entry.grid(row=3,column=1, padx=5, pady=5)

self.chk1.grid(row=6, column=0, padx=1, pady=1)

self.mutation_range_entry.grid(row=5, column=1, padx=5, pady=5)

self.chk2.grid(row=6, column=1, padx=1, pady=1)

self.chk3.grid(row=7, column=1, padx=1, pady=1)

self.chk4.grid(row=7, column=0, padx=1, pady=1)



self.submit_button = Button(text=" calculate ", command=self.gaInfo, font='arial 17',

bg='green', state='active', bd=5, height=1, width=30)


self.submit_button.grid(row=12, column=0, columnspan=3, pady=8)


self.root.mainloop()


def gaInfo(self):

try:

function = Func(self.f_entry.get())


optimizer = OptimizerGA(function)

optimizer.startGA( chromosomes_number=int(self.chromosomes_number_entry.get()),

generations_number=int(self.generations_number_entry.get()),

mutation=self.mutation.get(), mutation_range=self.mutation_range.get(),

optimizer=self.optimizer_entry.get(),

statistics=self.statistics.get(), save=self.save.get(), plot=self.plot.get())
```

```
if self.statistics.get():
with open('results/GA-statistics.txt', 'r') as f:
mytext = f.read()
root = Tk()

root.title("STATISTICS")
root.geometry('700x500+100+100')

frame = tk.Frame(master = root, bg = 'grey')
frame.pack(fill='both', expand='yes')
text = tkst.ScrolledText(
master = frame,
wrap = tk.WORD,
width = 700,
height = 500
)

text.pack()
text.insert(1.0, mytext)
root.mainloop()

except Exception as err:
print('Ошибка!\n', type(err))
print(err)
self.errorGUI()

def errorGUI(self):
self.root.destroy()
root = Tk()
```

```
root.title("ERROR")

root.geometry('300x150+100+100')

text = Text(width=20, height=3, font='arial 20', fg='red')

text.insert(1.0, "\nЧТО-ТО ПОШЛО НЕ ТАК!")

text.pack()

ok_button = Button(text=" OK ", command=root.destroy, font='arial 17',

bg='red', state='active', bd=3, height=2, width=8)

ok_button.pack()

root.mainloop()
```

```
def __init__(self):

self.root = Tk()

self.root.title("Settings")

self.root.geometry('550x300+100+100')

self.root.configure(bg='white')

# photo = PhotoImage(file = "logo.png")

# w = Label(self.root, image=photo)

# w.grid()

self.f = StringVar()

self.chromosomes_number = IntVar()

self.generations_number = IntVar()

self.optimizer = StringVar()

self.mutation = BooleanVar()

self.mutation_range = IntVar()

self.statistics = BooleanVar()
```

```
self.save = BooleanVar()

self.plot = BooleanVar()


self.chromosomes_number.set(4)

self.generations_number.set(10)

self.optimizer.set('min')

self.mutation.set(1)

self.mutation_range.set(2)

self.statistics.set(1)

self.plot.set(1)

self.save.set(0)


self.chk1 = Checkbutton(

text="Mutation", variable=self.mutation, onvalue=1, offvalue=0)

self.chk2 = Checkbutton(text="Show statistics",

variable=self.statistics, onvalue=1, offvalue=0)

self.chk3 = Checkbutton(text="Save all files",

variable=self.save, onvalue=1, offvalue=0)

self.chk4 = Checkbutton(

text="Show plot", variable=self.plot, onvalue=1, offvalue=0)


self.f_label = Label(

text="Input f(x, y) in python format:", font='arial 13')

self.chromosomes_number_label = Label(

text="Input individs number (multiple of 4):", font='arial 13')

self.generations_number_label = Label(

text="Input generations number:", font='arial 13')

self.mutation_range_label = Label(
```

```
text="Input mutation range:", font='arial 13')

self.optimizer_label = Label(

text="Input optimizer (min or max):", font='arial 13')


self.f_label.grid(row=0, column=0, sticky="w")

self.chromosomes_number_label.grid(row=1, column=0, sticky="w")

self.generations_number_label.grid(row=2, column=0, sticky="w")

self.optimizer_label.grid(row=3, column=0, sticky="w")

self.mutation_range_label.grid(row=5, column=0, sticky="w")


self.f_entry = Entry(textvariable=self.f)

self.chromosomes_number_entry = Entry(

textvariable=self.chromosomes_number)

self.generations_number_entry = Entry(

textvariable=self.generations_number)

self.mutation_range_entry = Entry(textvariable=self.mutation_range)

self.optimizer_entry = Entry(textvariable=self.optimizer)


self.f_entry.grid(row=0, column=1, padx=5, pady=5)

self.chromosomes_number_entry.grid(row=1, column=1, padx=5, pady=5)

self.generations_number_entry.grid(row=2, column=1, padx=5, pady=5)

self.optimizer_entry.grid(row=3, column=1, padx=5, pady=5)

self.chk1.grid(row=6, column=0, padx=1, pady=1)

self.mutation_range_entry.grid(row=5, column=1, padx=5, pady=5)

self.chk2.grid(row=6, column=1, padx=1, pady=1)

self.chk3.grid(row=7, column=1, padx=1, pady=1)

self.chk4.grid(row=7, column=0, padx=1, pady=1)


self.submit_button = Button(text=" calculate ", command=self.gaInfo, font='arial 17',
```

```
bg='green', state='active', bd=5, height=1, width=30)

self.submit_button.grid(row=12, column=0, columnspan=3, pady=8)

self.root.mainloop()

def gaInfo(self):
try:
function = Func(self.f_entry.get())

optimizer = OptimizerGA(function)
optimizer.startGA( chromosomes_number=int(self.chromosomes_number_entry.get()),
generations_number=int(
self.generations_number_entry.get()),
mutation=self.mutation.get(), mutation_range=self.mutation_range.get(),
optimizer=self.optimizer_entry.get(),
statistics=self.statistics.get(), save=self.save.get(), plot=self.plot.get())

if self.statistics.get():
with open('results/GA-statistics.txt', 'r') as f:
mytext = f.read()
root = Tk()

root.title("STATISTICS")
root.geometry('700x500+100+100')

frame = tk.Frame(master=root, bg='grey')
frame.pack(fill='both', expand='yes')
text = tkst.ScrolledText(
```

```
master=frame,
wrap=tk.WORD,
width=700,
height=500
)

text.pack()
text.insert(1.0, mytext)
root.mainloop()

except Exception as err:
print('Ошибка!\n', type(err))
print(err)
self.errorGUI()

def errorGUI(self):
self.root.destroy()
root = Tk()
root.title("ERROR")
root.geometry('300x150+100+100')
text = Text(width=20, height=3, font='arial 20', fg='red')
text.insert(1.0, "\nЧТО-ТО ПОШЛО НЕ ТАК!")
text.pack()
ok_button = Button(text=" OK ", command=root.destroy, font='arial 17',
bg='red', state='active', bd=3, height=2, width=8)
ok_button.pack()
root.mainloop()
```

```
@@ -2,10 +2,4 @@



if __name__ == '__main__':

gui = gaGUI()
```

```
gui = gaGUI()
```

22 geneticOptimizer/func.py

```
@@ -1,16 +1,16 @@

from numpy import sin, cos, tan,\

arcsin, arccos, arctan, arctan2,\

sinh, cosh, tanh,\

arcsinh, arccosh, arctanh,\

exp, expm1, exp2,\

log, log2, log10, log1p

arcsin, arccos, arctan, arctan2,\

sinh, cosh, tanh,\

arcsinh, arccosh, arctanh,\

exp, expm1, exp2,\

log, log2, log10, log1p



class Func:



def __init__(self, to_exec):
```

```
to_exec = to_exec.strip()

exec('self.f = lambda x, y: ' + to_exec)


class Func:

def __init__(self, to_exec):

to_exec = to_exec.strip()

exec('self.f = lambda x, y: ' + to_exec)

def __call__(self, x, y):

return self.f(x, y)

def __call__(self, x, y):

return self.f(x, y)
```

330 geneticOptimizer/optimizer.py

```
@@ -1,166 +1,176 @@

import numpy as np

import numpy.random as rd

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import numpy.random as rd

import pandas as pd

import matplotlib.pyplot as plt

import matplotlib.animation as animation

import tkinter.scrolledtext as tkst

import tkinter as tk

import tkinter as tk



from tkinter import *
```

```
from accessify import protected

from matplotlib import cm

from mpl_toolkits.mplot3d import Axes3D

from matplotlib.ticker import LinearLocator, FormatStrFormatter

from collections import OrderedDict

from tkinter import *

from accessify import protected

from matplotlib import cm

from mpl_toolkits.mplot3d import Axes3D

from matplotlib.ticker import LinearLocator, FormatStrFormatter

from collections import OrderedDict


import sys, os

import sys

import os


from func import Func



class OptimizerGA:

def __init__(self, function):

self.function = function


@protected

def generate_new_part(self, chromosomes, mutation=False, mutation_range=2, optimizer='min'):

values = [self.function(*chromosome) for chromosome in chromosomes]

chromosomesDict = dict(zip([str(i) for i in range(4)], values))


if optimizer == 'min':
```

```
chromosomesDict = OrderedDict(sorted(chromosomesDict.items(), key=lambda t: t[1]))

elif optimizer == 'max':

chromosomesDict = OrderedDict(sorted(chromosomesDict.items(), key=lambda t: -t[1]))

else:

raise ValueError(str(optimizer) + ' should be max or min')


chromosome_indexes = list()


for chromosome_index in chromosomesDict.keys():

chromosome_indexes.append(chromosome_index)


good_chromosome = chromosomes[int(chromosome_indexes[2])]

better_chromosome = chromosomes[int(chromosome_indexes[1])]

best_chromosome = chromosomes[int(chromosome_indexes[0])]


new_part = np.array([ [better_chromosome[0] + float(mutation_range * mutation * rd.rand(1) -
(mutation_range / 2)), best_chromosome[1]],

[good_chromosome[0] + float(mutation_range * mutation * rd.rand(1) - (mutation_range / 2)),
best_chromosome[1]],

[best_chromosome[0], better_chromosome[0] + float(mutation_range * mutation * rd.rand(1) -
(mutation_range / 2))],

[best_chromosome[0], good_chromosome[1] + float(mutation_range * mutation * rd.rand(1) -
(mutation_range / 2))]])

return new_part


@protected

def next_generation(self, mutation=False, mutation_range=2, optimizer='min'):

part = np.array([self.chromosomes[j] for j in range(0 , 4)])

new_population = self.generate_new_part(part, mutation, mutation_range, optimizer)


for parts_number in range(1, int(len(self.chromosomes) / 4)):
```

```
part = np.array([self.chromosomes[j] for j in range(parts_number * 4, (parts_number + 1) *
4)])

new_part = self.generate_new_part(part, mutation, mutation_range, optimizer)

new_population = np.append(new_population, new_part, axis=0)


return new_population


@protected

def calculate(self, optimizer='min'):

if optimizer == 'min':

return min([self.function(*chromosome) for chromosome in self.chromosomes])

elif optimizer == 'max':

return max([self.function(*chromosome) for chromosome in self.chromosomes])

else:

raise ValueError(optimizer + ' should be max or min')


@protected

def plotGA(self, chromosomes_number, generations_number, optimizer, save=False):

data = pd.concat([pd.read_csv('generations/generation_{}.csv'.format(i + 1), index_col=0)

for i in range(generations_number)], ignore_index=True)


data['time'] = [i for i in range(chromosomes_number * generations_number)]


def update_graph(num):

df = data[abs(num * chromosomes_number - data['time']) <= 2 * chromosomes_number]

graph.set_data(np.array(df['x']), np.array(df['y']))

graph.set_3d_properties(np.array(df['f(x, y)']))

title.set_text('generation={}'.format(num + 1))

return title, graph,
```

```
fig = plt.figure(figsize = (15, 8), num='GA animation')

ax = fig.add_subplot(111, projection='3d')



# Make data.

X = np.arange(-4, 4, 0.25)

Y = np.arange(-4, 4, 0.25)

X, Y = np.meshgrid(X, Y)

Z = self.function(X, Y)



# set colormap if it is needed

theCM = cm.get_cmap()

theCM._init()

alphas = np.abs(np.linspace(-1, 1, int(theCM.N)))

theCM._lut[:-3,-1] = alphas

# Plot the surface.

surf = ax.plot_surface(X, Y, Z, cmap=theCM,

linewidth=0, antialiased=True, alpha=0.6)

title = ax.set_title('GA-optimizer plot')



df = data[data['time'] == 0]

graph, = ax.plot(np.array(df['x']), np.array(df['y']), np.array(df['f(x, y)']),

linestyle="", c='black', marker='2', ms=2)
```

```
anim = animation.FuncAnimation(fig, update_graph, generations_number, interval=200,
save_count=True)


# Customize the z axis.

ax.set_zlim(-5, 5)

ax.zaxis.set_major_locator(LinearLocator(10))

ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))


ax.set_xlabel('X')

ax.set_ylabel('Y')

ax.set_zlabel('Z')


# Add a color bar which maps values to colors.

fig.colorbar(surf, shrink=0.8, aspect=3)


plt.show()

if save:

anim.save('results/GA-animation.gif', writer='imagemagick', fps=60)


def startGA(self, chromosomes_number=4, generations_number=10,

mutation=False, mutation_range=2, optimizer='min',

statistics=True, save=False, plot=True):

self.chromosomes = np.array([(mutation_range * rd.rand(2) - int(mutation_range / 2)) for i in
range(chromosomes_number)])


f = open('results/GA-statistics.txt', 'w')

for i in range(generations_number):

self.chromosomes = self.next_generation(mutation, float(mutation_range / (i + 1)), optimizer)
```

```
df = pd.DataFrame(self.chromosomes, columns=['x', 'y'])

x = np.array(df['x'])

y = np.array(df['y'])

df['f(x, y)'] = self.function(x, y)

df.to_csv("generations/generation_{}.csv".format(i + 1))



f.write('_' * 70)

f.write('\nINFO about generation {}:\n'.format(i + 1))

for chromosome in self.chromosomes:

f.write('chromosome {} gives value: {}\n'.format(chromosome, self.function(*chromosome)))

f.write('{} value for this generation: {}\n'.format(optimizer, self.calculate(optimizer)))


f.close()


if plot:

self.plotGA(chromosomes_number, generations_number, optimizer, save)


if not(save):

for i in range(generations_number):

os.remove("generations/generation_{}.csv".format(i + 1))


def __init__(self, function):

self.function = function


@protected

def generate_new_part(self, chromosomes, mutation=False, mutation_range=2, optimizer='min'):

values = [self.function(*chromosome) for chromosome in chromosomes]

chromosomesDict = dict(zip([str(i) for i in range(4)], values))
```

```
if optimizer == 'min':

chromosomesDict = OrderedDict(

sorted(chromosomesDict.items(), key=lambda t: t[1]))

elif optimizer == 'max':

chromosomesDict = OrderedDict(

sorted(chromosomesDict.items(), key=lambda t: -t[1]))

else:

raise ValueError(str(optimizer) + ' should be max or min')


chromosome_indexes = list()


for chromosome_index in chromosomesDict.keys():

chromosome_indexes.append(chromosome_index)


good_chromosome = chromosomes[int(chromosome_indexes[2])]

better_chromosome = chromosomes[int(chromosome_indexes[1])]

best_chromosome = chromosomes[int(chromosome_indexes[0])]


new_part = np.array([ [better_chromosome[0] + float(mutation_range * mutation * rd.rand(1) -
(mutation_range / 2)), best_chromosome[1]],

[good_chromosome[0] + float(mutation_range * mutation * rd.rand(

1) - (mutation_range / 2)), best_chromosome[1]],

[best_chromosome[0], better_chromosome[

0] + float(mutation_range * mutation * rd.rand(1) - (mutation_range / 2))],

[best_chromosome[0], good_chromosome[1] + float(mutation_range * mutation * rd.rand(1) -
(mutation_range / 2))]])

return new_part


@protected
```

```
def next_generation(self, mutation=False, mutation_range=2, optimizer='min'):

part = np.array([self.chromosomes[j] for j in range(0, 4)])

new_population = self.generate_new_part(

part, mutation, mutation_range, optimizer)


for parts_number in range(1, int(len(self.chromosomes) / 4)):

part = np.array([self.chromosomes[j] for j in range(

parts_number * 4, (parts_number + 1) * 4)])

new_part = self.generate_new_part(

part, mutation, mutation_range, optimizer)

new_population = np.append(new_population, new_part, axis=0)


return new_population


@protected

def calculate(self, optimizer='min'):

if optimizer == 'min':

return min([self.function(*chromosome) for chromosome in self.chromosomes])

elif optimizer == 'max':

return max([self.function(*chromosome) for chromosome in self.chromosomes])

else:

raise ValueError(optimizer + ' should be max or min')


@protected

def plotGA(self, chromosomes_number, generations_number, optimizer, save=False):

data = pd.concat([pd.read_csv('generations/generation_{}.csv'.format(i + 1), index_col=0)

for i in range(generations_number)], ignore_index=True)


data['time'] = [i for i in range(
```

```
chromosomes_number * generations_number)]

def update_graph(num):
df = data[abs(num * chromosomes_number - data['time'])
<= 2 * chromosomes_number]
graph.set_data(np.array(df['x']), np.array(df['y']))
graph.set_3d_properties(np.array(df['f(x, y)']))
title.set_text('generation={}'.format(num + 1))
return title, graph,

fig = plt.figure(figsize=(15, 8), num='GA animation')

ax = fig.add_subplot(111, projection='3d')

# Make data.
X = np.arange(-4, 4, 0.25)
Y = np.arange(-4, 4, 0.25)
X, Y = np.meshgrid(X, Y)
Z = self.function(X, Y)

# set colormap if it is needed
theCM = cm.get_cmap()
theCM._init()
alphas = np.abs(np.linspace(-1, 1, int(theCM.N)))
theCM._lut[:-3, -1] = alphas

# Plot the surface.
surf = ax.plot_surface(X, Y, Z, cmap=theCM,
linewidth=0, antialiased=True, alpha=0.6)
```

```
title = ax.set_title('GA-optimizer plot')


df = data[data['time'] == 0]

graph, = ax.plot(np.array(df['x']), np.array(df['y']), np.array(df['f(x, y)']),

linestyle="", c='black', marker='2', ms=2)


anim = animation.FuncAnimation(

fig, update_graph, generations_number, interval=200, save_count=True)


# Customize the z axis.

ax.set_zlim(-5, 5)

ax.zaxis.set_major_locator(LinearLocator(10))

ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))


ax.set_xlabel('X')

ax.set_ylabel('Y')

ax.set_zlabel('Z')


# Add a color bar which maps values to colors.

fig.colorbar(surf, shrink=0.8, aspect=3)


plt.show()

if save:

anim.save('results/GA-animation.gif', writer='imagemagick', fps=60)


def startGA(self, chromosomes_number=4, generations_number=10,

mutation=False, mutation_range=2, optimizer='min',

statistics=True, save=False, plot=True):
```

```
self.chromosomes = np.array(
[(mutation_range * rd.rand(2) - int(mutation_range / 2)) for i in range(chromosomes_number)])

f = open('results/GA-statistics.txt', 'w')
for i in range(generations_number):
self.chromosomes = self.next_generation(
mutation, float(mutation_range / (i + 1)), optimizer)
df = pd.DataFrame(self.chromosomes, columns=['x', 'y'])
x = np.array(df['x'])
y = np.array(df['y'])
df['f(x, y)'] = self.function(x, y)
df.to_csv("generations/generation_{}.csv".format(i + 1))

f.write('_' * 70)
f.write('\nINFO about generation {}:\n'.format(i + 1))
for chromosome in self.chromosomes:
f.write('chromosome {} gives value: {}\n'.format(
chromosome, self.function(*chromosome)))
f.write('{} value for this generation: {}\n'.format(
optimizer, self.calculate(optimizer)))

f.close()

if plot:
self.plotGA(chromosomes_number,
generations_number, optimizer, save)

if not(save):
for i in range(generations_number):
```

```
os.remove("generations/generation_{}.csv".format(i + 1))
```

17 geneticOptimizer/settingsGA.py

```
class gaParams:

def __init__(self, f: str,

chromosomes_number: int, generations_number: int,

mutation: bool ,optimizer: str):


self.f = f

self.chromosomes_number = chromosomes_number

self.generations_number = generations_number

self.mutation = mutation

self.optimizer = optimizer

def __init__(self, f: str,

chromosomes_number: int, generations_number: int,

mutation: bool, optimizer: str):


self.f = f

self.chromosomes_number = chromosomes_number

self.generations_number = generations_number

self.mutation = mutation

self.optimizer = optimizer
```

ДОДАТОК В ІНСТРУКЦІЯ КОРИСТУВАЧА

Оптимізатор навантаження на основі генетичних алгоритмів

Як встановити – треба встановити пакет geneOptimizer в env - це можна зробити, наприклад, за допомогою дій github, PyCharm або вручну з репозиторію github. Переконайтеся, що у папці генетичного оптимізатора / у вас є папки generations and results

Як користуватись

наберіть env/bin/python3 -m geneticOptimizer для запуску цієї програми та відкриття графічного інтерфейсу налаштувань з декількома полями:

f(x, y) - визначити фітнес функцію

15. individs number - встановити число індивидів
16. generations number - встановити число поколінь
17. mutation range - встановити діапазон мутаці

Крім того, необов’язкові поля:

mutation - додає мутацію до кожної частини нового покоління

- show statistics - показати точну інформацію про кожне покоління
- save all files - зберегти у статистиці geneOptimizer / результатів з анімацією у .gif та GA-Statistics.csv geneOptimizer / генерацій із колами x, y, f (x, y)
- show plot - показати анімацію еволюції

Про алгоритм

Ця програма приблизно розраховує мінімум або максимум функції з двома параметрами на основі генетичного алгоритму. Кожна особина - це одна хромосома (x, y), отже, особина та хромосома мають однакове значення; покоління складається з частин, де одна частина становить 4 особини, як наслідок, кількість особин кратна 4.

Генетичний алгоритм робить схрещування в кожній частині окремо після того, що додає його до покоління. Принцип кросоверу:

(x_better, y_best), (y_best, y_best), (x_best, y_better), (x_best, y_good)

де (x_best, y_best), (x_better, y_better), (x_good, y_good) вибираються окремо. Діапазон мутатуїнів визначає "потужність" мутованих генів: діапазон мутацій ~ 1 / число покоління

Планується:

Додайте варіанти мутації, такі як відсоток мутаційних генів, крок мутації, різновид мутації

Додайте параметри вибору на основі різних GA

Додайте більше показників для GA та більше інформації у статистиці, наприклад, час, придатність тощо.

Додайте спеціальний список функцій, які можуть проілюструвати, як працює пакет

ДОДАТОК Г – ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1 Загальні відомості

1.1 Позначення і найменування програми

Найменування: GA

Позначення: ПЗ розподілення навантаження.

1.2 Програмне забезпечення, необхідне для функціонування програми

Програмне забезпечення розподілення навантаження є крос-платформним. І основною необхідною вимогою для функціонування виробу є доступ в мережу.

1.3 Мови програмування

Програмне розподілення навантаження реалізовано на мові Python.

2 Функціональне призначення

2.1 Класи вирішуваних завдань і призначення програми

Програмне забезпечення розподілення навантаження повинно проводити виконання наведених нижче функцій:

— передбачуваність: роззподілення навантаження повинно проводитися згідно згідно з наданими критеріями відбору
— рівномірна завантаження ресурсів системи;
— масштабуємість: алгоритм повинен зберігати працездатність при збільшенні навантаженняправедливість: потрібно гарантувати, щоб на обробку кожного запиту виділялися системні ресурси і не допустити виникнення ситуацій, коли один запит обробляється, а всі інші чекають своєї черги;
— ефективність: всі сервери, які обробляють запити, повинні бути зайняті на 100%; бажано не допускати ситуації, коли один з серверів простоює в очікуванні запитів на обробку (відразу ж обмовимося, що в реальній практиці ця мета досягається далеко не завжди);
— скорочення часу виконання запиту: потрібно забезпечити мінімальний час між початком обробки запиту (або його постановкою в чергу на обробку) і його завершення;

— скорочення часу відгуку: потрібно мінімізувати час відповіді на запит користувача.

2.2 Функціональні обмеження

При завантаженню файлу зображення його розмір не повинен перевищувати 5 Мб

Т147

3 Опис логічної структури

3.1 Загальна структура

3.1.1 Дистрибутив ПЗ розподілення навантаження

ПЗ розподілення навантаження не розповсюджеється.

3.1.2 Логічна структура програми

Логічна стругтура розподілення навантаження мості відповідає логічній структурі

3.2 Алгоритми роботи прогрми

3.2.1 Запуск ПЗ розподілення навантаження проводиться здгідно інструкції з додатку Д

3.2.2 Графічна оболонка

Роботу з програмою описана в Додатоку Б

3.3 Використовувані методи

Використовується генетичний алгоритм

3.4 Зв'язок із стороннім ПЗ

ПЗможе запускатися з використанням cron.

4 Використовувані технічні засоби

Даний програмний продукт вимагає технічний засіб – персональний комп’ютер, ноутбук, планшет чи мобільний телефон з можливістю виходу в мережу інтернет.

Для ПЗ пред'являються такі апаратні

вимоги до ПЕОМ:

− процесор ARM або x86 від 1.3 GHz і вище;

− 1ГБ оперативної пам'яті та більше;

− можливість підключення до мережі інтернет.

Швидкість роботи ПЗ на конкретному комп'ютері залежить також від характеристик окремих його комплектуючих (процесора, оперативної пам'яті і ін.) та швидкості підключення до інтеренету.

5 Виклик і завантаження

Запуск ПЗ згідно інтсрукції користувача.

6 Вхідні і вихідні дані

Вхідні дані – перелік складових системи.

Вихідні дані – доповнена табліця марштуризації

Додаток Д- ПРОГРАМА І МЕТОДИКА ВИПРОБУВАНЬ

Об’єктом випробувань є програмне забезпечення для розподілення навантаження на основі генетичних алгоритмів

- Мета випробувань
- Перевірка функціональних можливостей програмного забезпечення на відповідність вимогам технічного завдання.

Вимоги до програми;

- передбачуваність: роззподілення навантаження повинно проводитися згідно згідно з наданими критеріями відбору
- рівномірна завантаження ресурсів системи;
- масштабуємість: алгоритм повинен зберігати працездатність при збільшенні навантаженняправедливість: потрібно гарантувати, щоб на обробку кожного запиту виділялися системні ресурси і не допустити виникнення ситуацій, коли один запит обробляється, а всі інші чекають своєї черги;
- ефективність: всі сервери, які обробляють запити, повинні бути зайняті на 100%; бажано не допускати ситуації, коли один з серверів простоює в очікуванні запитів на обробку (відразу ж обмовимося, що в реальній практиці ця мета досягається далеко не завжди);
- скорочення часу виконання запиту: потрібно забезпечити мінімальний час між початком обробки запиту (або його постановкою в чергу на обробку) і його завершення;
- скорочення часу відгуку: потрібно мінімізувати час відповіді на запит користувача.

- вимоги до програмної документації;

Склад програмної документації, що надається на випробування:

- Технічне завдання;
- Текст програми;

—Опис програми;

—Програма і методика випробувань;

—Результати проходження тестів;

—Інструкція користувача.

•Склад і порядок випробувань;

Розподілення навантаження на мережевому рівні передбачає вирішення наступного завдання: потрібно зробити так, щоб за одну конкретну IP-адресу сервера відповідали різні фізичні машини.

•

•Методи випробувань.

•

•match in on $ ext_if proto tcp to port 80 rdr-to $ web_servers round-robin sticky-address

pass in on $ int_if from $ lan_net \

route-to {($ ext_if1 $ ext_gw1), ($ ext_if2 $ ext_gw2)} \