Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ БАГАТОПОТОЧНОГО АЛГОРИТМУ ЕВОЛЮЦІЙНОГО ПОШУКУ
RESEARCH AND SOFTWARE IMPLEMENTATION OF MULTITHREADED EVOLUTIONARY SEARCH ALGORITHM

Виконав: студент(ка) 2 курсу, групи КНТ-219м
Спеціальності
122 Комп'ютерні науки
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Системи штучного інтелекту

Єрак Є.В.
(прізвище та ініціали)
Керівник Олійник А.О.
(прізвище та ініціали)
Рецензент Гофман Є.О.
(прізвище та ініціали)

2020

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(КОД І НАЙМЕНУВАННЯ)

Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ, д.т.н, проф.
_____С.О. Субботін
“____” _____20___року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Єрака Єгора Валерійовича
(прізвище, ім’я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація багатопоточного алгоритму еволюційного пошуку. Research and Software Implementation of Multithreaded Evolutionary Search Algorithm

керівник проєкту (роботи) Олійник Андрій Олександрович, к.т.н., доцент ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “ 3 ” листопада 2020 року № 301
2. Строк подання студентом проєкту (роботи) 15 грудня 2020 року
3. Вихідні дані до проєкту (роботи) технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. 6. Економіко-організаційна частина. 7. Охорона праці та безпека в надзвичайних ситуаціях.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5	Олійник А.О., доцент		
6	Пожуєва Т.О., професор		
7	Коробко О.В., ст. викладач		
Нормоконтролер	Калініна М.В., асистент		

7. Дата видачі завдання “ 4 ” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області.	1 тиждень	
2	Розробка та затвердження технічного	2 тиждень	Технічне завдання
	завдання.		
3	Проєктування структури системи.	3-4 тиждень	Структурна схема
4	Розробка схеми функціонування	5-6 тиждень	Функціональна
	системи та модулів програми.		схема. Модулі
5	Створення програмного коду системи.	7-8 тиждень	Текст програми
6	Тестування та відлагодження програми.	9 тиждень	Працездатна
			система
6	Тестування та відлагодження програми.	9 тиждень	Працездатна
			система
7	Розробка розділів пояснювальної	10 тиждень	ПЗ
	записки.		
8	Розробка слайдів презентації.	11-12 тиждень	Слайди презентації
9	Захист роботи.	13 тиждень	

Студент(ка) _____ Єрак Є.В.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Олійник А.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 120 с., 25 рис., 6 табл., 4 додатки, 35 джерел.

БАГАТОПОТОЧНИЙ АЛГОРИТМ, ЕВОЛЮЦІЙНИЙ МЕТОД, ОПТИМІЗАЦІЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ЦІЛЬОВА ФУНКЦІЯ.

Об'єкт дослідження – процес оптимізації багатовимірних функцій.

Предмет дослідження – еволюційні методи оптимізації.

Мета роботи – дослідження та розробка багатопоточного алгоритму еволюційного пошуку.

Матеріали, методи та технічні засоби: структурне та об’єктно-орієнтоване програмування, мова програмування C#, IBM-сумісний комп’ютер з монітором, клавіатурою та маніпулятором типу «миша», що функціонує під управлінням операційної системи Microsoft Windows.

Результати. Створено методи та програмні засоби для оптимізації багатовимірних функцій на основі еволюційного підходу. Розроблена програмна система може успішно експлуатуватися на IBM-сумісних ЕОМ під управлінням операційних систем Microsoft Windows та використовуватися для синтезу оптимізації багатовимірних функцій.

Висновки. Розроблено програмне забезпечення для оптимізації багатовимірних функцій на основі еволюційного підходу. Проаналізовано предметну область, розроблено технічне завдання для реалізації програми, обрано середовище розробки, розроблено структурну схему програми, виконано конструювання програмного забезпечення для оптимізації багатовимірних функцій на основі еволюційного підходу, здійснено тестування розробленого програмного забезпечення.

Галузь використання – підтримка процесів прийняття рішень, розпізнавання образів, діагностування на основі даних.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 120 pages, 25 figures, 6 tables, 4 appendixes, 35 sources.

MULTILREAD ALGORITHM, EVOLUTIONARY METHOD, OPTIMIZATION, SOFTWARE, TARGET FUNCTION.

Object of study is a process of multidimensional functions optimization.

Subject of study are evolutionary methods of optimization.

The purpose of the work is to study and develop a multithreaded evolutionary search algorithm.

Materials, methods, and tools: structured and object-oriented programming, programming language C#, computer with a monitor, keyboard and mouse manipulator operating under the control of the operating system.

Results. The method and software for multidimensional functions optimization based on the evolutionary approach are developed. The developed software system can successfully operate on IBM-compatible computers running Microsoft Windows operating system and can be used for multidimensional functions optimization.

Conclusions. Software for optimization of multidimensional functions based on the evolutionary approach has been developed. The subject area is analyzed, the technical task for program realization is developed, the development environment is chosen, the structural scheme of the program is developed, the software design for optimization of multidimensional functions on the basis of the evolutionary approach is executed, the developed software is tested.

The field of use is a support of decision-making processes, pattern recognition, data-based diagnosis.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ГА – генетичній алгоритм;

ПГА – паралельний генетичний алгоритм;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

ПП – програмний продукт.

ВСТУП

Для вирішення завдань прийняття рішень при діагностуванні на основі даних, розпізнаванні образів та прогнозуванні необхідно розв’язувати оптимізаційні задачі. Відомі методи оптимізації, як правило, є градієнтними, що пов’язано з вимогами диференційованості та неперервності цільових функцій, крім того такі методи є методами локального пошуку. Еволюційні методи, на відміну від градієнтних, не накладають умов на вигляд та параметри цільової функції та характеризуються механізмами виходу з областей локальних екстремумів [1]–[15].

Проте суттєвим недоліком таких методів є дуже низька швидкість оптимізації, що обумовлює необхідність розробки нових еволюційних методів з використанням багатопоточності обчислень, що дозволить зменшити часові затрати на процес оптимізації.

У ході виконання роботи було реалізовано метод еволюційного пошуку з використанням технології потоків на мові програмування C# з використанням технології .NET. Для більш повної демонстрації можливостей мови програмування, програму розроблено з використанням технології клієнт - сервер. Для максимального прискорення роботи алгоритму використовується блокуючий режим роботи сервера, коли сервер не обробляє запити декількох клієнтів одночасно. При реалізації багатопоточності використовується схема майстер-робітник, яка є не тільки найпростішою в реалізації, але й такою, що дає найбільший виграш при оптимізації складних функцій у зв’язку з тим, що не потребує додаткових ресурсів для зведення результатів роботи потоків.

Метою роботи є дослідження та розробка багатопоточного алгоритму еволюційного пошуку.

Для досягнення поставленої мети необхідно було розв’язати такі завдання:

– аналіз методів еволюційної оптимізації;

– розробка технічного завдання для реалізації програми;

– вибір засобів створення програмного забезпечення;

– розробка багатопоточного алгоритму еволюційного пошуку;

– розробка структурної схеми програми;

– конструювання програмного забезпечення реалізації багатопоточного алгоритму еволюційного пошуку;

– тестування розробленого програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Еволюційні алгоритми як методи оптимізації

Еволюційний алгоритм являє собою метод оптимізації, заснований на концепціях природного відбору і генетики. Як правило, до еволюційних алгоритмів відносять генетичні алгоритми (ГА). У цьому підході змінні, що характеризують рішення, представлені у вигляді генів в хромосомі. ГА оперує кінцевою безліччю рішень (популяцією) – генерує нові рішення, як різні комбінації частин рішень популяції, використовуючи такі оператори, як відбір, рекомбінація (кросинговер) і мутація. Нові рішення позиціонуються в популяції відповідно до їх становища на поверхні досліджуваної функції [1]–[15].

Ідею ГА підказала сама природа і роботи Ч. Дарвіна. Робиться наступне припущення: якщо взяти два цілком хороших виконання завдання і яким-небудь чином отримати з них нове рішення, то буде існувати висока ймовірність того, що нове рішення вийде хорошим або навіть кращим [1]–[15].

Для реалізації цього використовують моделювання еволюції (природного відбору), або якщо простіше – боротьби за виживання. У природі, за спрощеною схемою, кожна тварина прагне вижити, щоб залишити після себе якомога більше потомства. Вижити в таких умовах можуть лише найсильніші.

Тоді нам залишається організувати деяку середу – популяцію, населити її рішеннями - особинами, і влаштувати їм конкурентні умови. Для цього потрібно визначити функцію, за якою буде визначатися сила особини - якість запропонованого нею рішення. Ґрунтуючись на цьому параметрі можна визначити для кожної особини кількість нащадків, що залишаються нею або ймовірність того, що ця особина залишить нащадка. Причому, не виключений варіант, коли особина з дуже низьким значенням цього параметру помре.

Припустимо потрібно оптимізувати деяку функцію $F(X_1, X_2, .., X_n)$. Нехай ми шукаємо її глобальний максимум. Тоді, для реалізації ГА потрібно визначити, як ми будемо зберігати рішення. По суті, нам потрібно помістити всі X_1-X_n в деякий вектор, який буде грати роль хромосоми. Один з найбільш поширених способів – кодувати значення змінних у бітовому векторі. Наприклад, виділимо кожній вхідній змінній по 8 біт. Тоді вхідний вектор буде мати довжину $L = 8n$.

Нехай кожна особина складається з масиву X і значення функції F на змінних, витягнутих з цього масиву. Тоді ГА буде складатися з таких кроків [1]–[15]:

– генерація початкової популяції – заповнення популяції особинами, в яких елементи масиву X заповнені випадковим чином;

– оцінювання поточної популяції – кожна хромосома (особина, точка в просторі пошуку) оцінюється мірою її пристосованості відповідно до того, наскільки є гарним відповідне їй рішення задачі. Пристосованість визначається як обчислена цільова функція для кожної з хромосом;

– вибір батьківської пари – для цього часто може використовуватися оператор відбору, який може полягати, наприклад, у тому, що беруться K особин з максимальними значеннями функції F і складаються з них всі можливі пари (K * (K-1) / 2);

– кросинговер – беремо випадкову точку t на масиві X (0 .. L-1). Тепер, всі елементи масиву з індексами 0-t нової особини (нащадка) заповнюємо елементами з тими ж індексами, але з масиву X першої батьківської особини. Інші елементи заповнюються з масиву другої батьківської особини. Для другого нащадка робиться навпаки - елементи 0-t беруть від другого нащадка, а решта - від першого;

– нові особини з деякою вірогідністю мутують - інвертується випадковий біт масиву X цієї особини. Ймовірність мутації зазвичай вважають порядку 1%;

– отримані особи-нащадки додаються в популяцію після переоцінки; зазвичай нову особину додають замість найгіршою старої особини, за умови що значення функції на новій особини вище значення функції на старій (непристосованій) особині;

– якщо найкраще рішення в популяції нас не задовольняє, то переходимо на крок 2 (хоча найчастіше цієї умови немає, а ітерації ГА виконуються нескінченно).

Варто відзначити, що деякі реалізації ГА передбачають також такі кроки як відбір особин для розмноження і генерація пар з відібраних особин. При цьому кожна особина може бути задіяна в одній і більше парі, залежно від використовуваного алгоритму.

1.2 Переваги генетичного алгоритму

Приведемо наступні переваги генетичних алгоритмів [1]–[15]:

– вони не вимагають жодної інформації про поверхню відповіді;

– розриви, існуючі на поверхні відповіді мають незначний вплив на повну ефективність оптимізації;

– вони стійкі до потрапляння в локальні оптимуми;

– вони добре працюють при вирішенні масштабних проблем оптимізації;

– можуть бути використані для широкого класу задач;

– прості і прозорі в реалізації;

– можуть бути використані в задачах з мінливим середовищем.

1.3 Недоліки генетичного алгоритму

Не бажано і проблематично використовувати ГА [1]–[15]:

– у випадку, коли необхідно знайти точний глобальний оптимум;

– час виконання функції оцінки дуже великий;

– необхідно знайти всі рішення задачі, а не одне з них;

– конфігурація кодування рішення не є простою;

– для завдань швидкої локалізації одного оптимального значення;

– для завдань визначення кількох (або всіх) глобальних екстремумів.

Проте найсуттєвішим недоліком генетичних алгоритмів є дуже великий час їх роботи, що обумовлює необхідність модифікації існуючих та розробки нових еволюційних методів на основі паралельних обчислень.

1.4 Порівняльний аналіз генетичних алгоритмів

Останнім часом запропоновано багато різних генетичних методів і в більшості випадків вони є мало схожими на узагальнений генетичний метод. З цієї причини під терміном «генетичні методи» мають на увазі достатньо широкий клас методів, часом мало схожих один на одного [2]–[12].

Методи еволюційного пошуку можуть бути класифіковані за різними критеріям [1]–[15].

Укрупнену класифікаційну схему методів еволюційного пошуку наведено на рис. 1.1 [2]. У кожному з наведених на рис. 1.1 генетичних методів є можливим застосування різних генетичних операторів, тому варіанти генетичного пошуку, що розрізняються лише генетичними операторами, в схему не включено.

Нижче наведено класифікацію еволюційних методів [2]:

– за класом методів еволюційний пошук може бути розподілений на чотири групи: генетичні алгоритми, еволюційні стратегії, генетичне програмування та еволюційне програмування;

– за способом організації відтворення виділяють популяційні та стаціонарні еволюційні методи. Популяційний спосіб передбачає генерацію на кожній ітерації еволюційного пошуку всіх генотипів нової популяції P_{t+1} у відповідності до одного розподілу ймовірностей, обумовленого поточною популяцією P_t і розподілами ймовірностей операторів відбору, схрещування

й мутації. При стаціонарному способі відтворення на кожній ітерації створюється одна або дві нові особини, що заміщають у популяції особини з найменшим значенням фітнес-функції, або інші особини, обрані по деякому евристичному правилу;

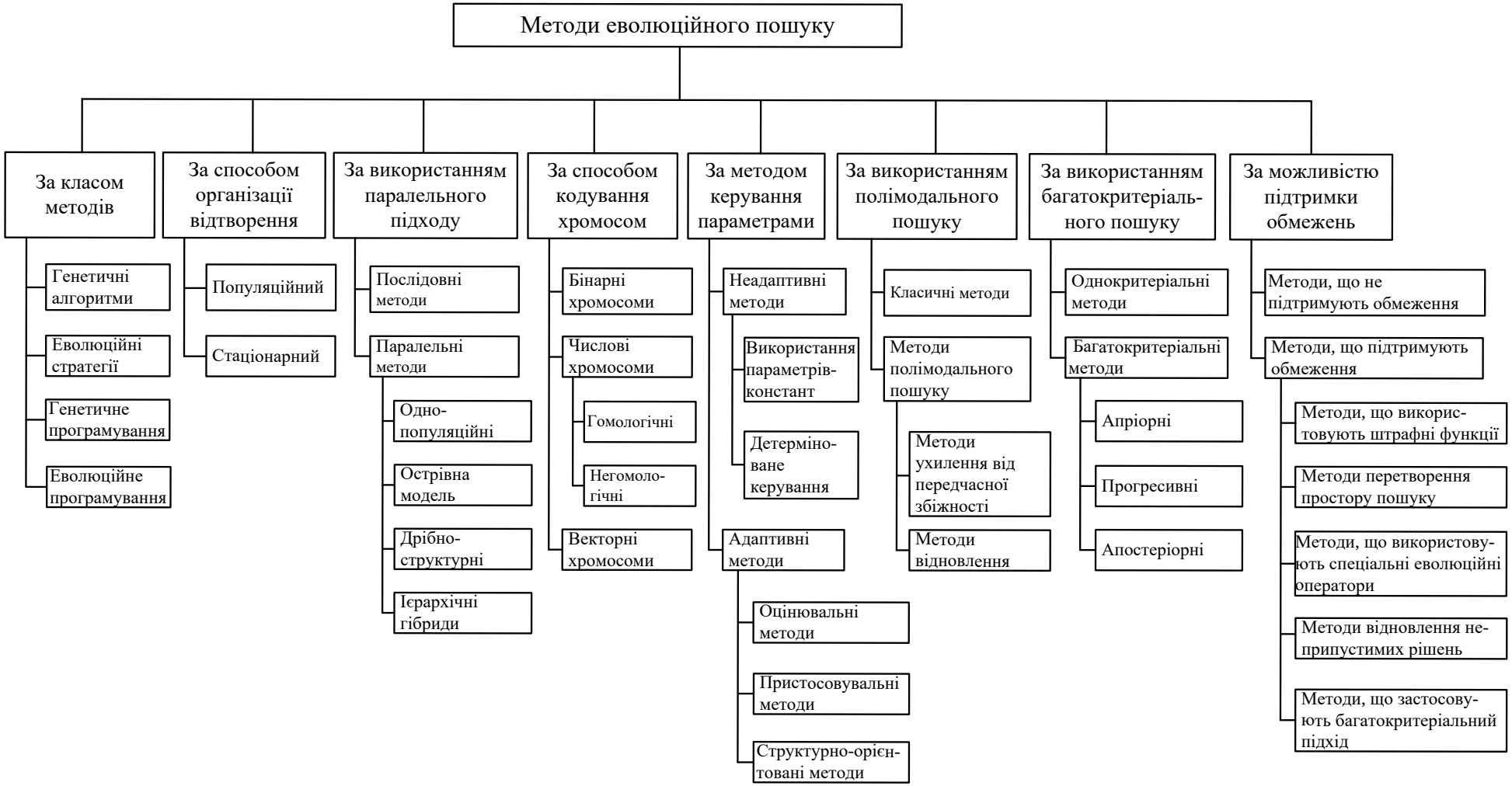


Рисунок 1.1 – Класифікація методів еволюційного пошуку [2]

– за використанням паралельного підходу методи еволюційного пошуку можуть бути класифіковані на послідовні й паралельні. Послідовні еволюційні методи використовують єдину популяцію хромосом на кожній ітерації. У випадку застосування паралельних методів на кожній ітерації використовується декілька підпопуляцій. До паралельних методів відносяться: однопопуляційні методи; острівна модель еволюційного пошуку; дрібноструктурні еволюційні методи; ієрархічні гібриди;

– за способом кодування хромосом виділяють еволюційні методи, що використовують бінарні хромосоми, числові хромосоми й векторні хромосоми. У бінарних хромосомах гени можуть приймати два значення. Числові хромосоми містять гени, що приймають будь-які значення в заданому інтервалі. При цьому в гомологічних числових хромосомах допускається наявність генів з однаковими значеннями, а в негомологічних хромосомах всі гени приймають різні значення. Векторні хромосоми складаються з генів, які являють собою вектор цілих чисел;

– за методом керування параметрами еволюційного пошуку виділяють неадаптивні методи й адаптивні методи. До неадаптивних методів керування параметрами відносяться: використання параметрів-констант; детерміноване керування параметрами. Адаптивні методи керування можуть бути класифіковані в такий спосіб: оцінювальні методи; пристосовувальні методи; структурно-орієнтовані методи (розбиття популяції й метод просторової популяційної структури);

– за використанням полімодального пошуку виділяють класичні методи й методи полімодального пошуку. Методи полімодального пошуку дозволяють виділити декілька оптимумів, розташованих у різних точках простору пошуку. До методів полімодального пошуку відносяться: методи ухилення від передчасної збіжності (методи уповільнення генетичної збіжності й методи запобігання появи співпадаючих рішень); методи відновлення (методи множинного заміщення, методи перезапуску і фазові методи);

– за кількістю критерії пошуку пошуку виділяють однокритеріальні та багатокритеріальні методи еволюційної оптимізації. Методи багатокритеріальної еволюційної оптимізації дозволяють виявити оптимум сукупності цільових функцій і розподіляються на: апріорні методи (метод агрегувальних функцій, лексикографічне впорядковування, мінімаксний метод, метод досягнення заданого значення); прогресивні методи; апостеріорні методи (популяційний підхід і пошук, заснований на підході Парето);

– за можливістю підтримки обмежень виділяють методи, що не підтримують обмеження, і методи, що підтримують обмеження. До методів, що підтримують обмеження, відносяться: методи, що використовують штрафні функції; методи перетворення простору пошуку; методи, що використовують спеціальні еволюційні оператори; методи відновлення неприпустимих рішень; методи, що застосовують багатокритеріальний підхід.

Відзначимо, що в умовах обмеженості часу та при необхідності оптимізації складних нелінійних функцій доцільно використовувати паралельні методи еволюційного пошуку.

1.5 Дослідження паралельних еволюційних методів

Схеми побудови паралельних генетичних алгоритмів (ПГА) діляться на два великі класи: острівний ПГА та ПГА типу «майстер-робітник» [1]–[15].

Схема острівного ПГА припускає, що відбувається розвиток декількох відносно незалежних популяцій. Вони проводять обмін кращими особинами через деяку кількість поколінь. Дана схема найбільш часто застосовується в кластерних архітектурах з пам'яттю. Ефективність схеми залежить від дуже великої кількості параметрів, серед яких [1]–[15]:

– топологія, яка визначає, які підпопуляції будуть вважатися сусідніми і, відповідно, між якими островами буде можливий обмін особинами;

– час ізоляції, який визначає, на протязі скількох епох ГА не проводитиметься міграція;

– ступінь міграції, яка визначає кількість особин, що беруть участь в міграції;

– стратегія відбору особин для міграції;

– стратегія видалення особин з підпопуляції при проведенні міграції;

– стратегія реплікації мігруючих особин.

Зауважимо також, що хоча популяції розвиваються незалежно і рівноправно, в реалізаціях такого виду алгоритмів виділяється сервер, який ініціалізує роботу і реалізує топологію обміну даними. Більше того, синхронізація роботи копій ГА на островах вимагає опису способу їх взаємодії. Точна і коректна реалізація такої взаємодії різних копій ГА в моделі островів є складним завданням. Для цього реалізуються протоколи, в яких фіксуються точки обміну інформацією між островами і сервером [1]–[15].

Модель «майстер-робітник» побудови ПГА реалізує тільки один цикл розвитку популяцій, який здійснюється на процесорі, який називається «майстер». На «робочі» процесори передається обчислювальне навантаження, яке пов'язане з оцінкою особин. Ця робота може бути виконана паралельно. Хоча організація паралельності за даною схемою вимагає деяких ресурсів, виграш може виявитися досить істотним. Особливо це буде помітно в тому випадку, якщо оцінка обчислюється за складними формулами, або на підставі моделювання Дана схема є більш простою і обрана нами для реалізації [1]–[15].

Також використовуються дрібноструктурні або дифузійні генетичні методи, які складаються з єдиної просторово структурованої популяції. Як правило, популяція подається у вигляді прямокутної сітки, у вузлах якої

розташовані особини. В ідеалі, на кожну особину виділяється окремий процесор для обчислення її фітнес-функції. Батьківську пару в даному методі можуть складати лише сусідні особини. Таким чином, характеристики кращої особини розповсюджуються по сітці не так швидко, як в інших методах, не дозволяючи тим самим деяким хромосомам домінувати на початкових ітераціях виконання генетичного пошуку.

У дворівневому генетичному методі DAGA2 реалізовано ідею багаторівневої еволюції: генетичний метод використовується на двох рівнях. На першому рівні DAGA2 виконує паралельний прогін генетичного методу (простий генетичний метод або будь-яка його модифікація), доки він не почне сходитися. Потім в найкращих рішеннях (хромосомах) кожної популяції визначаються найближчі один до одного об’єкти (гени), які утворюють кластери. На основі кластерів створюється нова хромосома як об’єкт для другого рівня DAGA2 [1]–[15].

DAGA2 може діяти як паралельний генетичний метод зі схрещуванням локальної області або на основі острівної моделі генетичного методу. DAGA2 діє як генетичний метод, що адаптується, в сенсі «виживання» найпристосованішого рішення [1]–[15].

Архітектура DAGA2 є аналогічною архітектурі генетичного методу «вприскування». У цій архітектурі хромосоми з підпопуляцій в грубій (неточній) моделі задачі мігрують в інші підпопуляції в негрубій (точній) моделі задачі.

Архітектура DAGA2 може бути модифікована на будь-яке число рівнів залежно від пам’яті й часових обмежень на отримання результату.

Таким чином, можна зробити висновок, що при реалізації багатопоточної роботи ГА алгоритму доцільно використовувати схему «майстер-робітник», оскільки така модель є досить простою для реалізації та більш ефективною у порівнянні з іншими моделями, особливо при роботі зі складними функціями.

1.6 Висновки за розділом 1

Проаналізовано еволюційні методи розв’язання завдання оптимізації багатовимірних нелінійних функціональних залежностей. Відзначено, що еволюційні методи, як і інші методи оптимізації, можуть зациклюватися в областях локальних екстремумів і вимагають значних ресурсів комп’ютеру та часу для виконання оптимізаційного процесу.

Визначено, що в умовах обмеженості часу та при необхідності оптимізації складних нелінійних функцій доцільно використовувати паралельні методи еволюційного пошуку. Досліджено паралельні еволюційні методи. Відзначено, що при реалізації багатопоточної роботи еволюційних методів доцільно використовувати схему «майстер-робітник», оскільки така модель є досить простою для реалізації та більш ефективною у порівнянні з іншими моделями, особливо при роботі зі складними функціями.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Розробка модифікованого паралельного генетичного алгоритму

У роботі як базис для розробки багатопоточного алгоритму еволюційного пошуку використано модель «майстер-робітник» паралельного еволюційного алгоритму. В результаті аналізу виявлено, що при даній схемі основні затрати ресурсів системи при реалізації алгоритму йдуть на обчислення значень (оцінювання) цільової функції.

Тому для пришвидшення роботи методу у роботі було запропоновано використовувати подання цільових функцій на основі бінарних дерев. При програмній реалізації запропонованої модифікації багатопоточного алгоритму еволюційного пошуку необхідно розробити модуль, що проводить синтаксичний аналіз формули, введеної користувачем у довільному вигляді.

При такому розрахунку формули, що представляє собою цільову функцію еволюційної оптимізації, цей процес можливо розбити на потоки та реалізувати на паралельній системі, що дозволить суттєво пришвидшити процес еволюційного пошуку.

Застосовуючи бінарне дерево, гілки якого мають значення або заданого числа, або номеру змінної у виразі за яких встановляться ціле число та елементами дерева, що мають посилання на подібні елементи, які відображають ліву та праву частину ріння.

Приклад бінарного дерева подання функціональної залежності наведено на рис. 2.1.

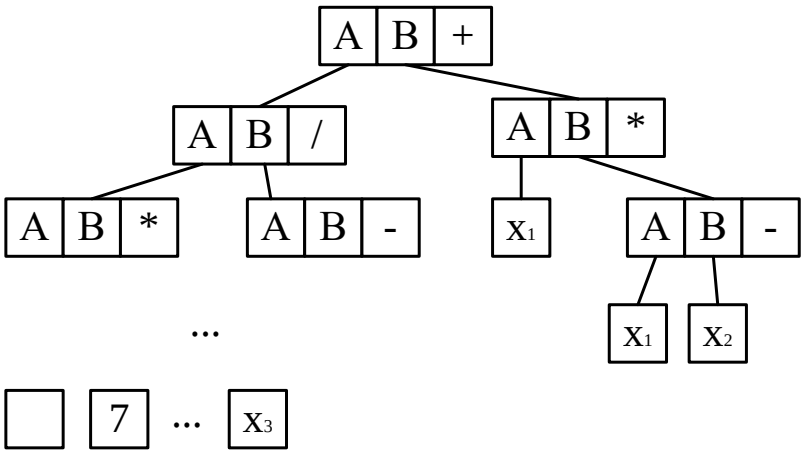


Рисунок 2.1 – Приклад бінарного дерева подання функціональної залежності

Таким чином, наукова новизна роботи полягає у тому, що набув подальшого розвитку багатопоточний алгоритм еволюційного пошуку на основі моделі «майстер-робітник», в якому використовується подання цільових функцій на основі бінарних дерев, що дозволяє розпаралелити етап обчислення цільових функцій та суттєво пришвидшити процес еволюційної оптимізації.

2.2 Аналіз технічного завдання

Основою для розробки програми являється технічне завдання, яке наведено в додатку А.

Результатом повинні бути програмний комплекс з легкого клієнту та серверу. Програма повинна мати зручний інтерфейс.

Клієнт має виконувати такі функції:

– надавати користувачеві можливість ввести усі необхідні для роботи дані;

– підтримувати зв'язок з сервером через протокол TCP;

– інтерфейс клієнту має бути зручний та інтуїтивно зрозумілий;

– реалізовувати вивід інформації, що надіслана сервером;

Сервер має використовувати наступні функції:

– реалізація ПГА;

– зв'язок з клієнтом у блокуючему режимі;

– синтаксичній аналіз формули, заданої строковим рядком та побудови бінарного дерева розрахунку.

2.3 Вибір інструментів для розробки системи

Для розроблення програмного забезпечення, що реалізує багатопоточний алгоритм еволюційного пошуку, було обрано мову програмування C# [16]–[18].

Корпорацією Microsoft запропоновано компонентно-орієнтований підхід до програмування на основі ідеології .NET, який є розвитком об'єктно-орієнтованого спрямування. Відповідно до цього підходу, інтеграція об'єктів (можливо, гетерогенної природи) проводиться на основі інтерфейсів, що представляють ці об'єкти (або фрагменти програм) як незалежні компоненти. Такий підхід суттєво полегшує написання та взаємодія програмних компонент у гетерогенному середовищі проектування та реалізації. Стандартизується зберігання і повторне використання компонент програмного проекту в умовах розподіленого мережевого середовища обчислень, де різні комп'ютери і користувачі обмінюються інформацією, наприклад, взаємодіючи в рамках дослідницького або бізнес-проекту [16]–[18].

Суттєвими особливостями мови програмування C#, які визначають її переваги для використання в даному проекті [16]–[18]:

– компонентна орієнтованість;

– код зібраний воєдино (декларації і реалізації об'єднані разом);

– уніфікована система типів і їх безпечність;

– автоматична і мануальна робота за пам'яттю;

– використання єдиної бібліотеки класів – CLR [16].

Крім того, важливим є те, що мова програмування C# підтримує реалізацію паралельних обчислень.

Таким чином, інструментарієм для розробки системи була обрана мова C#. Програму розроблено у FormApplication на базі технологія Net Framework.

За допомогою цих засобів можна створити зручний інтерфейс користувача, забезпечити достатню швидкодію програми та максимально прискорити розробку.

Виконаємо порівняння таких програмних рішень:

– Microsoft C# .Net 4.0 (Windows 7);

– Oracle Java SE Version 6 Update 24 (Windows 7);

– Oracle Java SE Version 6 Update 24 (Linux 2.6.35.27 Ubuntu 10.10);

– Novell Mono 2.11 C# (Linux 2.6.35.27 Ubuntu 10.10).

Порівняння швидкості різних мов програмування на математичних функціях та при обробці масивів різних типів наведено на рис. 2.2 та рис. 2.3.

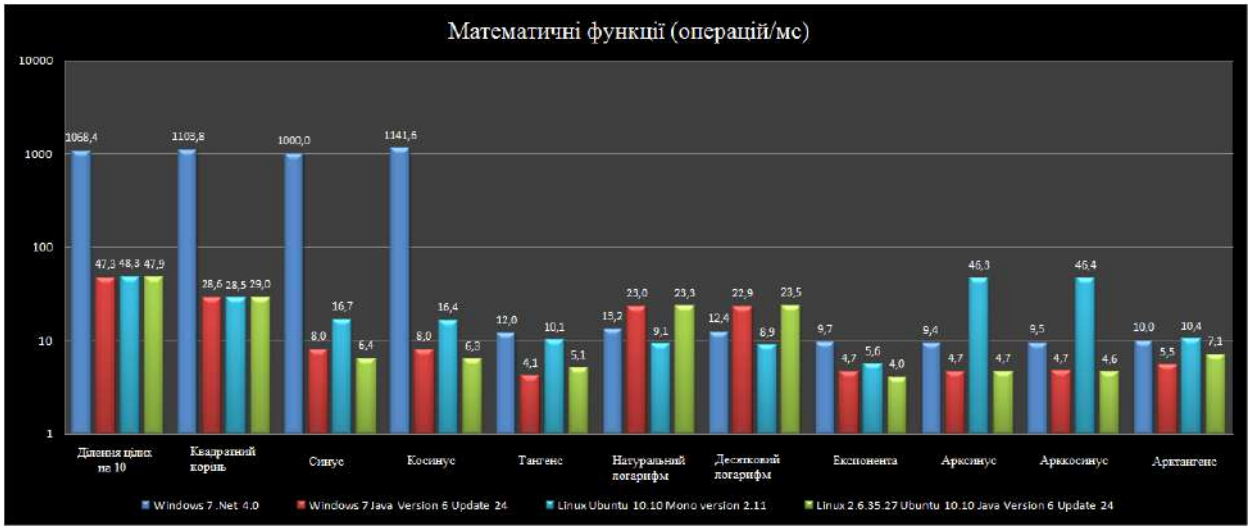


Рисунок 2.2 – Порівняння швидкості різних мов програмування на математичних функціях

З огляду на те що в першу чергу програми будуть працювати з масивами даних та математичними функціями доцільно використовувати мову C#, швидкість виконання обчислень за допомогою якої майже у два рази вище у порівнянні з іншими мовами.

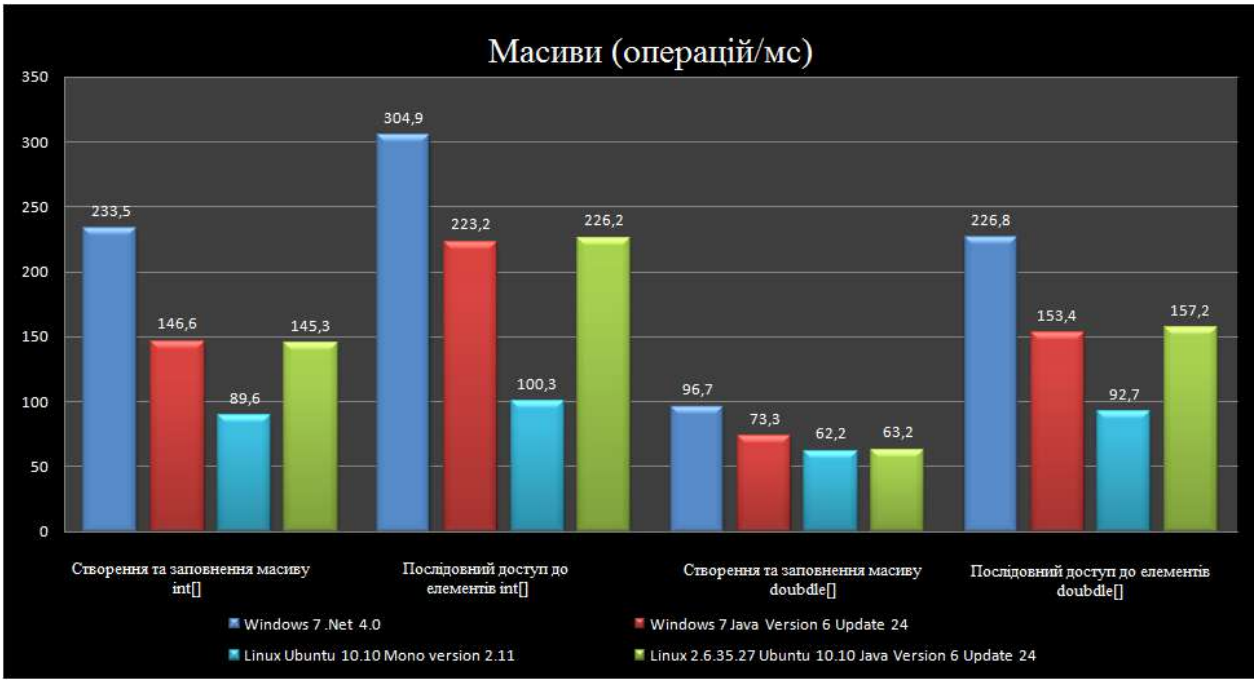


Рисунок 2.3 – Порівняння швидкості при обробці масивів різних типів

Порівняльний аналіз різних мов програмування наведено у табл. 2.1.

Таблиця 2.1 – Порівняльний аналіз мов програмування

Критерії порівняння	Мова програмування		
	C#	Java	C++
Паралельні обчислення	+	+	±
Швидкість розробки	++	+	+
Розроблення графічного інтерфейсу користувача	+	+	+
Швидкість реалізації операцій на математичних функціях	±	+	++
Швидкість реалізації операцій при обробці масивів різних типів	–	+	++

Таким чином, аналізуючи інформацію з табл. 2.1 та рис. 2.2-2.3, стає очевидним, що для програмної реалізації багатопоточного алгоритму еволюційного пошуку доцільно обрати мову програмування C#, яка ефективно підтримує реалізацію паралельних обчислень. Крім того,

швидкість виконання обчислень за допомогою цієї мови майже у два рази вище у порівнянні з іншими мовами.

2.4 Висновки за розділом 2

Набув подальшого розвитку багатопоточний алгоритм еволюційного пошуку на основі моделі «майстер-робітник», в якому використовується подання цільових функцій на основі бінарних дерев, що дозволяє розпаралелити етап обчислення цільових функцій та суттєво пришвидшити процес еволюційної оптимізації.

Для програмної реалізації багатопоточного алгоритму еволюційного пошуку обрано мову програмування C#, оскільки швидкість виконання обчислень за допомогою цієї мови майже у два рази вище у порівнянні з іншими мовами. Крім того, мова C# ефективно підтримує реалізацію паралельних обчислень, що є важливим при програмуванні багатопоточних алгоритмів.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Структура програмного забезпечення реалізації багатопоточного алгоритму еволюційного пошуку

Програма реалізована у вигляді клієнт-серверної моделі, на базі об’єктно-орієнтованого програмування.

Структурну схему програми наведено на рис. 3.1.

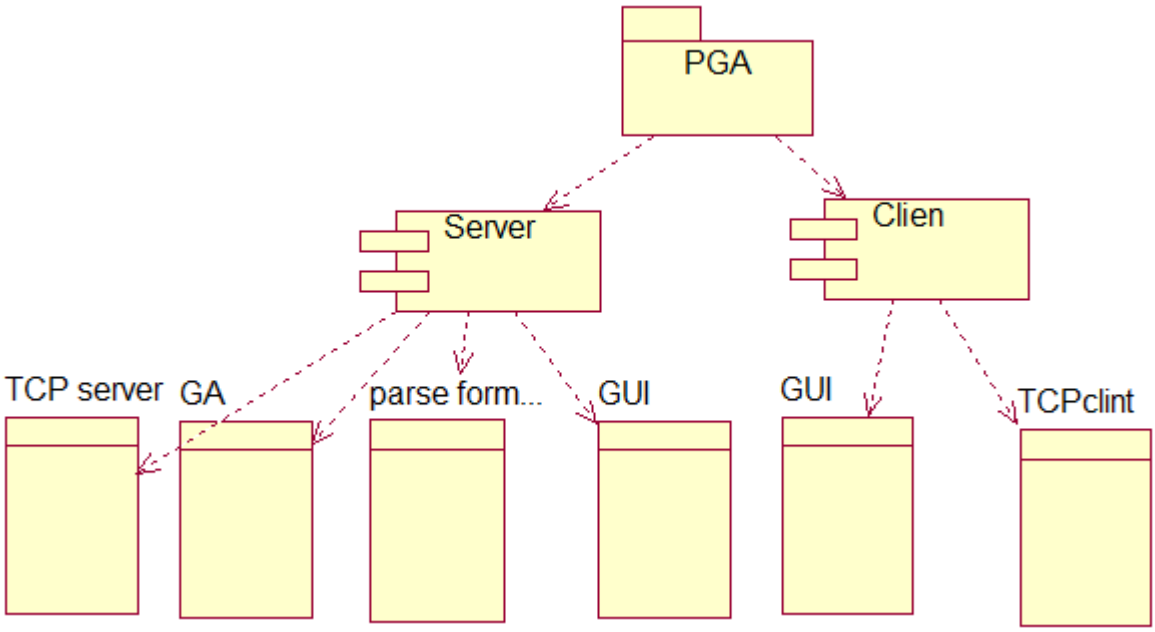


Рисунок 3.1 – Структурна схема програми

Клієнт надає користувачеві зручну можливість заповнити, форми для вводу даних, та надає доступ до результату роботи серверу. Має зручний та ергономічний інтерфейс з підказками для користувача та наданням користувачеві доступу до серверу з використанням мережевої технології, протоколу TCP.

Підпрограма серверу реалізує основні функції системи та все розрахунки. Для підвищення швидкості роботи сервер працює у блокуючому

режимі, тобто одночасно може обробляти запит лише з одного клієнту. Складається з 3 основних модулів та модулю інтерфейсу:

– модуль реалізації TCP з'єднання з клієнтом у блокуючому режимі;

– модуль синтаксичного аналізу формул, що аналізує символьній рядом та будує бінарне дерево за яким розраховується фітнес функції;

– модуль реалізації ПГА, що виконує основні функції програмами;

– модуль інтерфейсу що надає можливість адміністратору сервера, вибрати порт там запустити, або спинити сервер.

Стандартний .NET клас Form використовується у якості основного контролеру.

3.2 TCP Модуль клієнту

Мета цього модуля полягає у підтримці зв'язку клієнта з сервером, використовуючи коректні по відношенню сокети (реалізовані у вигляді спеціальних класів C#) та коректне по відношенню до WindiowsForm використання потоків.

```
namespace ClientApp
{
   public delegate void getDataClHendler(object sender, string str); //тип делегата для собітия получения сообщения
   class Client
   {
      public event getDataClHendler getData;        //события получения сообщения

      private StreamWriter swSender;              //поток для записив клана
      private StreamReader srReceiver;            //поток для чтения из канал
      private TcpClient tcpServer;               //идентификатор для тсп сервера
      private string strBuff;                   //полученное сообщение
      Thread tLisen;
      private bool flagOn = true;               //флаг сведетльствующий о соеденении клиента с сервером
      public  Client(string IPadr, int port)
      {
         try                                  //блок проверющий операции в случаее елси какято из операций пройдет неудачно будет вызванно исключение
         {
         IPAddress ipAddr = IPAddress.Parse(IPadr);     //получаем из конструктора IP адрес сервера
```

```
                tcpServer = new TcpClient();                //выделяем память под клас класс отвечающие за сервер в соеденении
                tcpServer.Connect(ipAddr, port);            //подключаемся к серверу
                srReceiver = new System.IO.StreamReader(tcpServer.GetStream()); //перенаправляем поток для стения данных сервера
                swSender = new System.IO.StreamWriter(tcpServer.GetStream()); //перенаправляем пook для отправки сообщений серверу
                tLisen = new Thread(lisen);                 //инициализируем поток ожидания сообщения от сервера
                tLisen.Start(); //запускаем поток ожидания сообщения от сервера
                }
                catch
                {
                   flagOn = false;
                   MessageBox.Show("Сервер не найден");

                }


             }

             void lisen()                                   //метод юспользуемый в качесве поток для ожидания и получения сообщения от сервера
             {
                try
                {
                   while ((strBuff = srReceiver.ReadLine()) != "" && flagOn == true)  //ждем сообщения от сервера если соеденение активно и сервер не прислал пустую строку
                   {
                      if (strBuff == null)
                      {

                      }
                      else
                      {
                         onGetMsg(strBuff);  //вызываем события получения сообщения
                      }

                   }
                }
                catch
                {
                   stop();
                }

             }
             void onGetMsg(string buff)  //описываем события получения сообщения где выполняем функцию записанную в делегат
             {
                getData(null,buff);
             }
             public void sendMsg(string buff) //отправка сообщения сервером
```

```
        {
            if (flagOn == true)
            {
                swSender.WriteLine(buff);
                swSender.Flush();
            }
        }
        public void stop()          //особождения ресурсво в случаи выключения клиента
        {
            if (flagOn)
            {
                tcpServer.Close();
                srReceiver.Close();
                swSender.Close();
                tLisen.Abort();
                flagOn = false;
            }
        }
    }
}
```

3.3 TCP Модуль серверу

Метою цього модуля є підтримка зв'язку сервера з клієнтом, використовуючи коректні по відношенню сокети (реалізовані у вигляді спеціальних класів C#) та коректні по відношенню до WindiowsForm використання потоків.

```
namespace ServerApp
{
    public delegate void getDataHendler(object sender, string str); //тип делегата для события получения сообщения
    class Server
    {

        public event getDataHendler getData; //создания события
        TcpClient tcpClient;                //запускаем поток прослушивания порта
        private StreamReader srReceiver;     //поток для отправления данных(в строковом формате)
        private StreamWriter swSender;       //поток для получеия данных (в строковом формате)
        private string strResponse;          //полученное сообщение
        private bool serverOn = false;       //флаг запуска  сервера
        private bool conetc = false;         //флаг установки соеденения
        private TcpListener tlsClient;       //прослушиванеи порта
        Thread tLisen;                      //идентификатор для поток для прослушки порта
```

```
        public Server(string IPadr, int port)
        {
            try
            {
                tlsClient = new TcpListener(port); //инициализация прослушки порта
                tlsClient.Start();              //запуск прослушки порта
                serverOn = true;                //поднимаем флаг запуска сервера
                tLisen = new Thread(lisen);           //инициализация потока прослушки
порта
                tLisen.Start();                     //запуск потока прослушки порта



            }
            catch
            {
                serverOn = false;       //в случаее неудачи болка трай опускаем флаг
                MessageBox.Show("Не удалось запустить сервер, возможно порт занят");
            }

        }
         private void lisen()
        {
            try
            {
                tcpClient = tlsClient.AcceptTcpClient();  //запускаем просшулку порта и
получаем ссылку на клиент
            }
            catch
            {
                tlsClient.Stop();
            }
            conetc = true;                          //поднимаем флаг получения клиента
            srReceiver = new System.IO.StreamReader(tcpClient.GetStream());
//перенаправляем поток чтения
            swSender = new System.IO.StreamWriter(tcpClient.GetStream());
//перенаправляем поток записи
            while ((strResponse = srReceiver.ReadLine()) != "" && serverOn)  //получаем
сообщения
            {
                // If it's invalid, remove the user
                if (strResponse == null)  //если сообщения есть
                {

                }
                else //если сообщения есть запускаем собітия получения сообщения  и
отдаем туда сообщения
                {
                    // Otherwise send the message to all the other users
                    onGetMsg(strResponse);
                }
            }
```

```
                }
                 public void stop();  //отключаем червер высвобождая все использованные
классы(освобождаем порт)
                 void onGetMsg(string buff);  //сообщения получения сообщения
                 public void sendMsg(string buff)  //метод отправления сообщения
                 {
                    swSender.WriteLine(buff);  //записываем в буфер строку(буффер+символ
конца строки)
                    swSender.Flush();          //особождаем поток в канал(передаем данные в
сесть)
                 }
              }
           }
```

3.4 Модуль синтаксичного аналізу формул

Цей модуль повинен по строковому рядку побудувати дерево, за допомогою якого буде проводитись розрахунок формули. Побудова дерева та розрахунок значення формули відбуваються за методом пошуку в глибину. Модуль розрахунку формули реалізований на основі патерну сінглтон для пришвидшення роботи системи.

```
              class FuncGraf
              {
               int _znak;// 0 - "-", 1 - "+", 2-"/", 3 - "*"
               double _value;
               int _argIndex;
               int _type; //0 - value 1- arg
               static  string _error = "OK";
               static double[] _args;
               FuncGraf _a;  // левая ветвь дерва
               FuncGraf _b;  // правая ветвь дерева
               public FuncGraf(double value) //конструктор привнивающий ветве знаения в
виде числа
                {
                   _value = value;
                   _type = 0;
                   _error = "OK";

                }
              public FuncGraf(int index) // конструктор присваювающий елменту значение
індекса перемнной
```

```
        {
            _argIndex = index - 49;
            _type = 1;
            _error = "OK";

        }

        public FuncGraf(int znak, FuncGraf a, FuncGraf b) // конструктор присваевающий елменту значения виражения из правой и левой ветві
         {
            _znak = _znak;
            _a = a;
            _b = b;
            _type = 2;
            _error = "OK";
         }
        public FuncGraf(string buff); // коструктор реалізующий первічний ввод формули і аналіз віржения


        public void setArgs(int count, double[] args) //функция задющия аргументи
        {
            _args = new double[count];
            for (int i = 0; i <= count-1; i++)
            {
                _args[i] = args[i];
            }
        }
        public void setArgs( double arg)
        {
            _args = new double[1];
            _args[0] = arg;

        }
        public void setArgs(double arg, double arg1)
        {
            _args = new double[2];
            _args[0] = arg;
            _args[1] = arg1;

        }
        public void setArgs(double arg, double arg1, double arg2)
        {
            _args = new double[3];
            _args[0] = arg;
            _args[1] = arg1;
            _args[2] = arg2;

        }
```

```
        public double calculate() //функция проізводяшая расчет виражения по заднним аргументам
        {
            if (_error == "OK")
            {
                if (_type == 0) return _value;  //возваршет  значения числа елси данній елемнт дерва ветвь
                else
                    if (_type == 1) return _args[_argIndex];  //возваршет  значения числа аргумент елси данній елемнт дерва ветвь
                else
                {
                    if (_b == null) return _a.calculate();
                    else
                    {
                        switch (_znak)  //реурсивній расчте текущего елмента по левой и правой ветве и занку между ними
                        {
                            case 0: return _a.calculate() - _b.calculate();

                            case 1: return _a.calculate() + _b.calculate();
                            case 3: return _a.calculate() * _b.calculate();
                            case 2:
                                {
                                    double b = _b.calculate();
                                    if (b != 0)
                                    {
                                        return _a.calculate() / _b.calculate();
                                    }
                                    else
                                    {
                                        _error = "error / by 0";
                                        return 0;
                                    }
                                }
                        }
                    }
                }
            }
            else
            {
                return 0;
            }
            return 0;
        }


    }
```

3.5 Модуль паралельного еволюційного пошуку

У цьому модулі реалізовано процеси паралельного еволюційного пошуку.

```
class GA
   {
      int _generationCount; // количество генераций
      double _eps;          // порог
      int _individualCount; // колічесво особей в пуляции
      Funk _funk = Funk.Instance;  // получение екземпляра синлтона формули
      Population _currentPop; // текушая популяция
      static Random rnd = new Random(System.Environment.TickCount);
      public GA(int generationCount, double eps, int individualCount, out double[] args,
out double result) // конструткор запускающий генетыческий алгоритм
      {
         _generationCount = generationCount;
         _individualCount = individualCount;
         _eps = eps;
         int i=0;
         int flag = 1;
         double fitnes = start(); // первий шаг генрация начальной виборки
         if (fitnes > eps)
         {
            flag = 0;
         }
         while (flag == 1) // цикил в котором пока не будет привішен лимит итераци
й или достигунт нужній фитнес ведтся работа алгоритма
         {
            fitnes = step(); // щаг алгоритма
            if (fitnes > eps)
            {
               flag = 0;
            }
            if (i == generationCount)
            {
               flag = 0;
            }
            i++;

         }
         // вивод результата
         result = fitnes;
         args = _currentPop.getFirstAgs();
      }
      private double start();
      private double step()
      {
```

```
                Population newPop = new Population(_currentPop.Count); // создане новой
популяцыи
                for (int j = 0; j < _currentPop.Count; j++) // выбор особей для селекцыи
                {

                   int perent1 = rnd.Next(_currentPop.Count / 2, _currentPop.Count - 1);
                   int perent2 = rnd.Next(_currentPop.Count / 2, _currentPop.Count - 1);
                   int maxRand=0;
                   while (perent1 == perent2 || maxRand ==4)
                   {
                      perent2 = rnd.Next(_currentPop.Count / 2, _currentPop.Count - 1);
                      maxRand++;
                   }
                   newPop.newPersonByIndex(j, _currentPop[perent1], _currentPop[perent2]);//
создания новой особи
                }
                newPop.sort(); // расчет фитнеса и сортировка по нему популяцыи
                _currentPop = newPop; // присвоение текущей поуляцыи значение новой
                return _currentPop.getFitness();
             }


          }
```

3.6 Тестування розробленого програмного забезпечення

Для демонстрації роботи програми спочатку був запущений сервер на порті 777.

Далі був запущений клієнт з наступними даними:

– налаштування мережі: IP: 127.0.0.1; port: 777;

– формула та її параметри: формула: ($x1+x2*x3$); мінімальне значення параметрів: 10; максимальне значення параметрів: 20; кількість змінних 3.

– налаштування алгоритму: кількість ітерацій :100; кількість екземплярів у популяції: 15; потрібне значення цільової функції: 100.

З налаштувань мережі видно, що тестовий запуск програм проводиться на одному комп’ютері.

Формула задана доволі проста, щоб було легше відстежити роботу ГА, та програмного комплексу в цілому.

З налаштувань алгоритму ми бачимо доволі велику кількість ітерацій та екземплярів в популяції. При доволі низькому фітнесі функції тобто закінчитись експеримент повинен через досягнення потрібного фітнесу.

Після натиснення клавіші “Send” ми бачимо результати знаходження максимум функції, змінні, при яких результат був досягнутий, та час роботи (рис. 3.2).

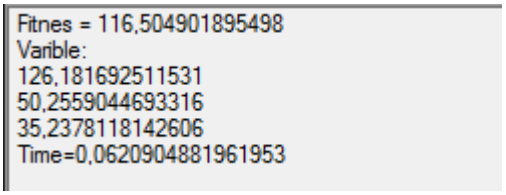


Рисунок 3.2 – Результати роботи програми

Проаналізувавши результати ми бачимо, що розрахунок зупинився при досягненні потрібного значення цільової функції.

У ході роботи було реалізовано багатопоточний алгоритм еволюційного пошуку на базі технологій .NET. Схема майстер - робітник передбачає використання фіксованої кількості додаткових потоків, що дорівнює кількості екземплярів у популяції, тому що їх цільові функції вираховуються одночасно.

Для тестування системи було використано комп’ютер на базі INTEL CORE i5 2400 x4 3.1GHz та з використанням операційній системі Microsoft Windows.

Порівнювати виграш від використання потоків доцільно, оцінюючи виграш в швидкості у порівняні із простим ГА. Також треба використовувати досить складну формулу. Не зважаючи на багатопоточність операційної системи, реальна кількість одночасно виконуваних потоків не може перевищувати кількість процесорів (чи ядер).

Графік прискорення паралельних еволюційних методів наведено на рисунку 3.3.

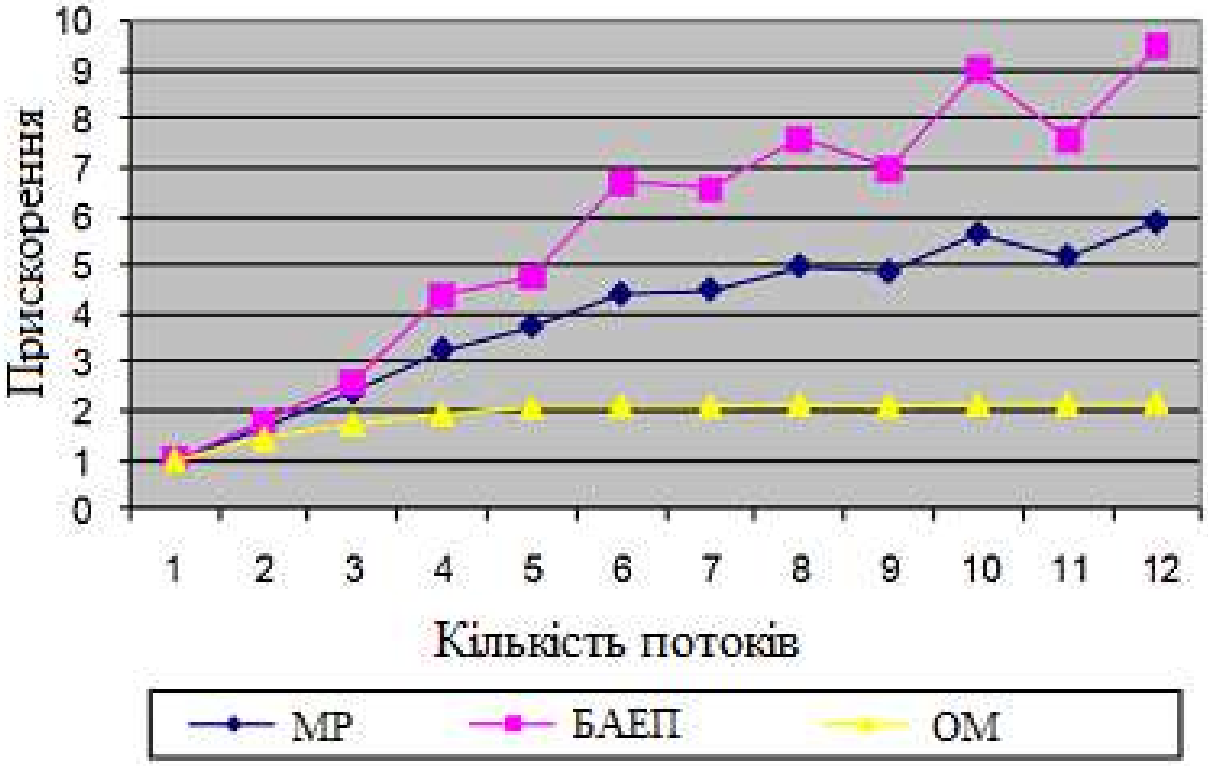


Рисунок 3.3 – Графік прискорення паралельних еволюційних методів

На рисунку 3.3 використовуються такі позначення: МР – паралельний еволюційний метод на основі моделі «майстер – робітник»; БАЕП – багатопоточний алгоритм еволюційного пошуку; ОМ – острівна модель еволюційного пошуку.

Як видно з рисунку, прискорення багатопоточного алгоритму еволюційного пошуку є кращим у порівнянні з іншими моделями (базової моделі «майстер – робітник» та острівної моделі).

Таким чином, запропонований багатопоточний алгоритм еволюційного пошуку дозволяє суттєво (в рази) прискорити процес еволюційної оптимізації у порівнянні з послідовними еволюційними методами. Графік прискорення свідчить про більш високу ефективність застосування розробленого багатопоточного алгоритму еволюційного пошуку у порівнянні з іншими паралельними реалізаціями еволюційних методів, зокрема методу на основі базової моделі «майстер – робітник» та острівної моделі еволюційного пошуку.

3.7 Висновки за розділом 3

Розроблено структурну схему програми. Програма реалізована у вигляді клієнт-серверної моделі, на базі об’єктно-орієнтованого програмування. Клієнт надає користувачеві зручну можливість заповнити, форми для вводу даних, та надає доступ до результату роботи серверу. Має зручний та ергономічний інтерфейс з підказками для користувача та наданням користувачеві доступу до серверу з використанням мережевої технології, протоколу TCP. Підпрограма серверу реалізує основні функції системи та всі розрахунки.

Виконано тестування розробленого програмного забезпечення. Виявлено, що запропонований багатопоточний алгоритм еволюційного пошуку дозволяє суттєво (в рази) прискорити процес еволюційної оптимізації у порівнянні з послідовними еволюційними методами. Графік прискорення свідчить про більш високу ефективність застосування розробленого багатопоточного алгоритму еволюційного пошуку у порівнянні з іншими паралельними реалізаціями еволюційних методів, зокрема методу на основі базової моделі «майстер – робітник» та острівної моделі еволюційного пошуку.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови застосування програми

Програмний продукт призначений для оптимізації цільових багатовимірних функціональних залежностей.

Для коректної роботи програму потрібно запускати в Windows 7 або більш пізнішій версії операційної системи від Microsoft з .NET.

Програма не вимоглива до процесора та оперативної пам’яті, тому запуститься і швидко працюватиме на будь якому комп’ютері з операційною системою Windows з використанням мінімуму ресурсів. Проте повністю її потенціал може розкритися лише на багатопроцесорних комп’ютерах.

Для експлуатації програмного комплексу необхідні такі програмно-технічні засоби:

– ПК під керівництвом операційної системи;

– тактова частота процесора 2 ГГц або більше, кількість ядер – 2 або більше;

– обсяг оперативної пам'яті не менше 2 Гб;

– обсяг вільного простору на диску - не менше 100 Мб;

– дозвіл екрана монітора не менше 1024x768;

– маніпулятор миша.

4.2 Характеристики програми

Програмні модулі мають назву, ServApp, ClienApp.

Розмір програмного забезпечення становить порядка 100 Мб на дисковому просторі.

Програмній комплекс має один режим стандартної роботи.

Програма повністю контролює оператора від помилок.

4.3 Звернення до програми

Розроблене програмне забезпечення поширюється у вигляді інсталяційного пакета. Для можливості роботи з програмою необхідно розмістити файли модулів програмного забезпечення на сервері. Перед використанням програми її необхідно встановити. Для цього необхідно запустити програму setup.exe, яка знаходиться на установчому диску. У відповідь на запит установчої програми потрібно вказати шлях встановлення програми. Після цього програма скопіює необхідні файли, створить і помістить в головне меню системи ярлик для запуску програми і зробить необхідні зміни в файлі реєстрації.

Після завершення процедури установки програму можна використовувати. Програму можна запустити будь-якими засобами операційної системи користувача.

Для запуску програми необхідно запустити файл "ServApp.exe",”ClienApp.exe”.

4.4 Вхідні і вихідні дані

Вхідними даними для програми є характеристики, за якими буде проводитись розрахунок: розмір популяції, кількість поколінь (ітерацій розрахунків), кількість вимірів, цільова функція.

Вихідними даними для програми є результати розрахунків: оптимальні значення цільової функції та значення вхідних параметрів, при яких досягаються відповідні значення цільової функції.

4.5 Повідомлення

Реалізована програмна система гарантує її стійке функціонування. У випадку виникнення помилок в програмі передбачено відображення

стандартних діалогових вікон. Подібні повідомлення можуть виникати при деяких небажаних ситуаціях, до яких можна віднести введення користувачем некоректних даних.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Програмний продукт призначений для оброблення даних при оптимізації цільових багатовимірних функціональних залежностей.

Для коректної роботи програму потрібно запускати в Windows 7 або більш пізнішій версії операційної системи від Microsoft з .NET.

Програма не вимоглива до процесора та оперативної пам’яті, тому запуститься і швидко працюватиме на будь якому комп’ютері з операційною системою Windows з використанням мінімуму ресурсів. Проте повністю її потенціал може розкритися лише на багатопроцесорних комп’ютерах.

Інтерфейс користувача забезпечує просту і ефективну взаємодію користувача з програмним продуктом, а також передбачає можливість його адаптації до вимог конкретного користувача.

5.2 Умови виконання програми

Для експлуатації програмного комплексу необхідні такі програмно-технічні засоби:

– ПК під керівництвом операційної системи;

– тактова частота процесора 2 ГГц або більше, кількість ядер – 2 або більше;

– обсяг оперативної пам'яті не менше 2 Гб;

– обсяг вільного простору на диску - не менше 100 Мб;

– дозвіл екрана монітора не менше 1024x768;

– маніпулятор миша.

Час роботи програми залежить від конфігурації комп'ютера, на якому вона виконується.

У процесі роботи програма виводить інформацію у вигляді:

– форми з вхідними та вихідними даними;

– форми з результатами розрахунків;

– інші форми.

5.3 Виконання програми

Розроблене програмне забезпечення поширюється у вигляді інсталяційного пакета. Для можливості роботи з програмою необхідно розмістити файли модулів програмного забезпечення на сервері. Перед використанням програми її необхідно встановити. Для цього необхідно запустити програму setup.exe, яка знаходиться на установчому диску. У відповідь на запит установчої програми потрібно вказати шлях встановлення програми. Після цього програма скопіює необхідні файли, створить і помістить в головне меню системи ярлик для запуску програми і зробить необхідні зміни в файлі реєстрації.

Після завершення процедури установки програму можна використовувати. Програму можна запустити будь-якими засобами операційної системи користувача.

Для запуску програми необхідно запустити файл "ServApp.exe","ClienApp.exe".

5.4 Приклад роботи з програмою

Після запуску програми на стороні сервера на екрані з’являється форма завантаження даних (рисунок 5.1).

На даній формі треба ввести порт, на якому буде працювати сервер. Також з цієї форми можна запустити та зупинити сервер, ввести шлях та підключити логи.

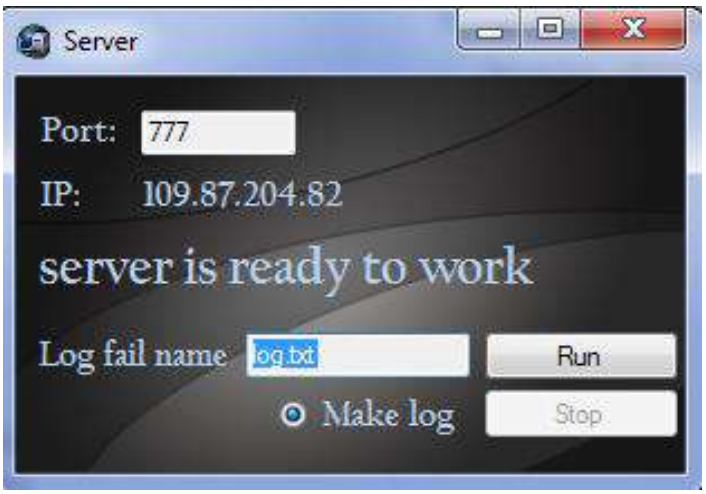


Рисунок 5.1 – Форма серверу

Для зручності адміністратору сервер виводить зовнішній IP комп'ютера. А по середні форми є статус – напис, що детально описує стан серверу.

Після запуску програми клієнта на екрані з'являється форма завантаження даних (рисунок 5.2).

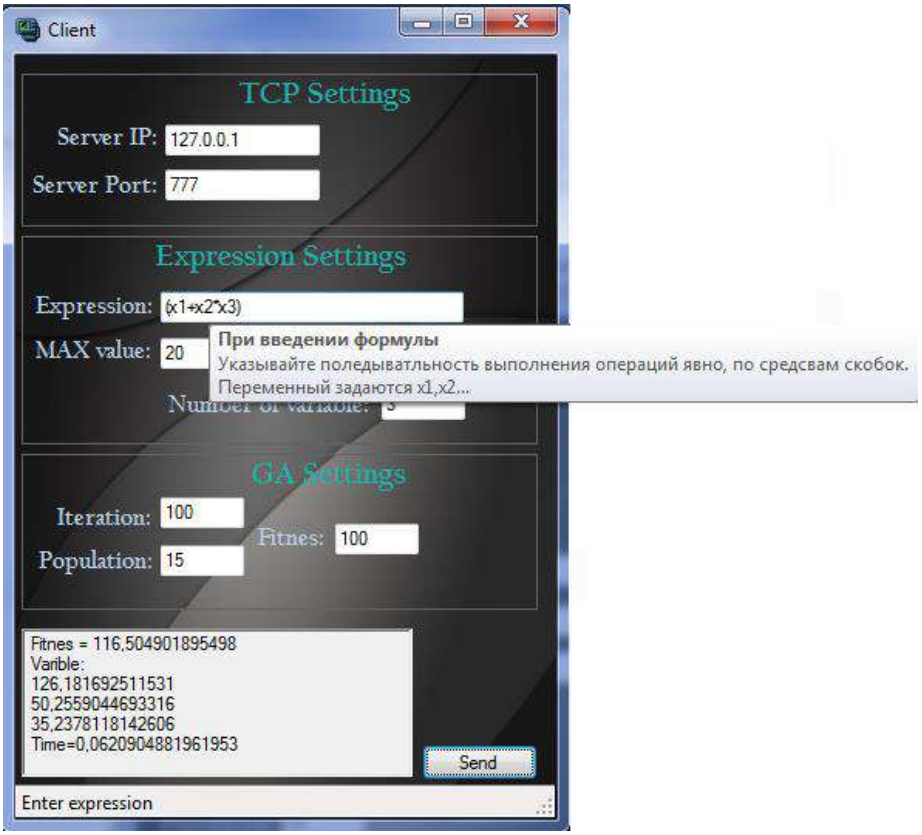


Рисунок 5.2 – Форма клієнту

Клієнт має зручний інтерфейс з усіма можливими підказками, та захистом від несправних дій користувача.

Клієнт являє собою 3 панелі вводу даних. Це панель налаштувань мережі (IP адрес й порт сервера), панель введення формули та її налаштувань (сама формула вводитися як рядок, при умові, що порядок операцій потрібно явно задавати скобками), панель налаштувань роботи алгоритму, де клієнт має задати кількість ітерацій, кількість екземплярів у популяції та потрібне значення цільової функції.

Після того, як усі данні будуть задані, користувач може за допомогою кнопки Send відправити запит на сервер. Результат запиту буде виведено у текстовому полі.

5.5 Повідомлення оператору

Як відзначено вище, у випадку виникнення помилок в програмі передбачено відображення стандартних діалогових вікон. Подібні повідомлення можуть виникати при деяких небажаних ситуаціях, до яких можна віднести введення користувачем некоректних даних.

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

У дипломній кваліфікаційній роботі магістра досліджено та реалізовано багатопоточний алгоритм еволюційного пошуку.

Впровадження в експлуатацію розробленого програмного забезпечення на виробничому підприємстві дозволить скоротити 0,75 ставки одного інженера з аналізу даних з заробітною платою 10 тис. гривень на місяць.

У розробці беруть участь програміст із заробітною платою 8000 грн. на місяць і консультант із заробітною платою 9000 грн. на місяць. На розробку проекту було витрачено 2 місяці. Розробка ПЗ розпочалася першого вересня і має бути виконано до кінця жовтня 2020 року.

6.1 Планування розробки програмного продукту

Весь комплекс розробки програмного продукту можна розділити на етапи [19]. Для кожного етапу вказуються трудомісткість, кількість виконавців і тривалість робіт. Інформація з характеристики робіт по розробці програми зведена у таблицю 6.1.

За даними таблиці 6.1 складається графік планування розробки програмного продукту [19].

Таблиця 6.1 – Характеристика робіт по розробці програми

Найменування робіт	Трудомісткість		Виконавці	Тривалість, днів
	люд-дн	%		
1	2	3	4	5
Аналіз ПЗ	10	15,3	Програміст Консультант	5
Визначення вимог до ПП	10	15,3	Програміст Консультант	5

Продовження таблиці 6.1

1	2	3	4	5
Розробка схеми функціонування програми	5	7,69	Програміст	5
Створення програмного коду	10	15,3	Програміст	10
Тестування і налагодження програмного продукту	5	7,69	Програміст	5
Складання програмної документації	5	7,69	Програміст	5
Разом	65	100		45

6.2 Визначення витрат на розробку програми

Для визначення витрат на розробку програми складають калькуляцію кошторисної вартості робіт, що включає статті:

– основна заробітна плата;

– додаткова заробітна плата;

– відрахування на єдиний соціальний внесок (ЄСВ);

– матеріали та комплектуючі вироби;

– витрати на спеціальне обладнання (на оплату машинного часу ЕОМ для написання і налагодження програмних засобів);

– накладні витрати.

Витрати за статтею «Основна заробітна плата» складаються з планового фонду зарплати всіх категорій працівників, зайнятих в розробці програми. Розрахунок зарплати на підставі даних про трудомісткості, представлений в таблиці 6.2.

Таблиця 6.2 – Основна заробітна плата

Посада виконавця	Чисельність, чол.	Місячний оклад	Кількість місяців роботи	Сума ЗП, грн.
Програміст	1	8000	2	16000
Консультант	1	9000	1	9000
Разом	2			25000

Додаткову заробітну плату приймають рівною 10% від основної зарплати. Вона розраховується за формулою (6.1):

$$ЗП_{дод} = ЗП_{осн} \cdot 10\%. \quad (6.1)$$

Підставивши величину основної заробітної плати в дану формулу отримаємо:

$$ЗП_{дод} = 25000 \cdot 0,1 = 2500 \text{ грн.}$$

Відрахування на єдиний соціальний внесок (ЄСВ)визначають у відсотковому відношенні від сум основної та додаткової зарплати з урахуванням премій і доплат. Відрахування вираховуються за формулою (6.2):

$$ОТ = (ЗП_{осн} + ЗП_{дод}) \cdot 0,22. \quad (6.2)$$

$$ОТ = (25000 + 2500) \cdot 0,22 = 6050 \text{ грн.}$$

Використовується два найменування матеріалів: картридж – 900 грн, папір - 100 грн. Витрати на матеріали розраховують за формулою (6.3):

$$M = \sum_{i=1}^{n} (Ц_i \cdot N_i \cdot (1 + K_{т.з.}) - Ц_{io} \cdot N_{io}), \quad (6.3)$$

де М – витрати на матеріали, покупні напівфабрикати і комплектуючі вироби, грн .;

$K_{т.з}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

$Ц_i$ – ціна і-го найменування матеріалу, напівфабрикату і комплектуючого, грн .;

N_i – потреба в і-му матеріалі, напівфабрикаті і комплектуючому;

$Ц_{io}$ – ціна зворотних відходів і-го найменування матеріалу, грн;

N_{io} – кількість зворотних відходів і-го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих.

Обчислимо:

$$Ц_{io} = 0;\ N_{io} = 0;\ K_{т.з} = 0{,}05;$$

$$М=(1 + 0{,}05) \cdot (900+100)=1050 \text{ грн.}$$

Разом витрати на матеріали становлять 1050 гривні.

У статтю «Витрати на спеціальне обладнання» входять витрати на придбання, транспортування, монтаж і налагодження нестандартного обладнання. Практично, в даному випадку, в цій статті враховуються витрати на оплату машинного часу ЕОМ для написання і налагодження програмних засобів. Для цього необхідно скласти кошторис «витрат на утримання і експлуатацію устаткування», виходячи з якого визначиться вартість одного машино-години роботи ПК, після множення якої на машинний час пішло на написання і налагодження програми отримаємо витрати на оплату машинного часу.

Амортизаційні відрахування визначають за формулою (6.4):

$$A = БВ \cdot \frac{H_a}{100}, \qquad (6.4)$$

де *БВ* – балансова вартість обчислювальної техніки, грн;

H_a – норма амортизаційних відрахувань на повне оновлення обчислювальної техніки, %.

Балансова вартість обчислювальної техніки становить 20000 грн. Тому амортизаційні відрахування становлять:

$$A = 20000 \cdot 0,25 = 5000 \text{ грн.}$$

Статтю "Експлуатація обладнання" розраховують підсумовуванням витрат на силову електроенергію і допоміжні матеріали. Витрати на електроенергію визначають за формулою (6.5):

$$C_э = N_н \cdot Ф_{эф} \cdot K_{зв} \cdot K_{зм} \cdot Ц_э, \quad (6.5)$$

де $N_н$ – номінальна потужність ЕОМ, кВт;

$Ф_{эф}$ – річний ефективний фонд часу роботи ЕОМ, машино-год;

$K_{зв}$ – середній коефіцієнт завантаження за часом;

$K_{зм}$ – коефіцієнт завантаження по потужності;

$Ц_э$ – ціна одного кВт· год електроенергії, грн ./ (кВт· год).

Номінальна Потужність ЕОМ – 0,2 кВт. Річний ефективний фонд часу роботи ЕОМ ставити 1800 годин. Середні коефіцієнти завантаження за часом та за потужністю рівні відповідно 0,9 та 0,6. Ціна однієї кіловат–години електроенергії ставить 2,11 грн. Отримуємо:

$$Ce = 0,2 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,11 = 410,18 \text{ грн.}$$

Приймаємо, що Се = 410 грн.

Витрати за статтею "Заробітна плата обслуговуючих робітників та відрахування на соціальне страхування в інші фонди" визначають за формулою (6.6):

$$ЗП_{обсл} = ФЗП_{г} \cdot (1 + К_{відрах}) \cdot \frac{t_{обсл}}{Ф_{еф.обсл}}, \quad (6.6)$$

де $ФЗП_{г}$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн.;

$К_{відрах}$ – коефіцієнт, що враховує відрахування на соціальне страхування і в інші фонди;

$t_{обсл}$ – час протягом року, необхідне технічне обслуговування ЕОМ, годин / рік. Приймаємо $t_{обсл} = 24$ год.;

$Ф_{еф.оюсл}$– річний ефективний фонд часу обслуговуючого персоналу, годин / рік. Приймаємо, що річний ефективний фонд часу обслуговуючого персоналу становить 1800 годин

Заробітна плата обслуговуючого персоналу = 6000 грн/місяць.

$$ФЗП_{г} = 6000 \cdot 12 = 72000 \text{ грн.}$$

$$ЗП_{обсл} = 72000 \cdot (1+0,22) \cdot 24/1800 = 1171 \text{ грн.}$$

Суму витрат за статтею "Поточний ремонт обладнання" обчислимо як 3% від балансової вартості обладнання, оскільки неможливо заздалегідь визначити, скільки коштів необхідно витрачати на придбання запасних частин наявного обладнання. Балансова вартість обчислювальної техніки становить 20000 грн. Тому сума витрат за статтею "Поточний ремонт обладнання" становить: $20000 \cdot 0,03 = 600$ грн.

Інших витрат за проектом не передбачено, тому сума витрат за цією статею дорівнює нуль гривень.

Кошторис витрат на утримання і експлуатацію устаткування наведений в таблиці 6.3.

Таблиця 6.3 – Кошторис витрат на утримання і експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація обладнання	5000
Заробітна плата основна і додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	1171
Експлуатація обладнання (крім витрат на поточний ремонт)	410
Поточний ремонт обладнання	600
Інші витрати	0
Разом	7181

Витрати на оплату машинного часу ЕОМ для написання і налагодження програмних засобів визначаються за формулою (6.7):

$$C_{мо} = P_{екс} \cdot t_{мо}, \tag{6.7}$$

де $C_{мо}$ – витрати на оплату машинного часу, грн;

$P_{екс}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн. / машино–год.;

$t_{мо}$ – машинний час цифрової ЕОМ для написання і налагодження даного програмного продукту, машино–год.

Отримуємо:

$$P_{екс} = 7181/1800 = 3{,}99 \text{ грн. / машино-год.}$$

Для написання і налагодження даного програмного продукту ЕОМ експлуатується протягом 45 днів в одну зміну по 8 годин, що складає в сумі 360 годин. Отримуємо:

$$С_{мо} = 3{,}99 \cdot 360 = 1436 \text{ грн.}$$

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і т.п.), утримання та експлуатацію будівель. Накладні витрати включаються до вартості розробки програми непрямим шляхом – у відсотках до основної заробітної плати розробників. Для обчислення цієї статті видатків у дипломної кваліфікаційної роботі приймемо, що накладні видатки становлять 50% від основної заробітної плати розробників, що складає 25000 грн. Тому сума накладних видатків становить: 25000·0,5=12500 грн.

Калькуляція кошторисної вартості робіт з розробки програми наведена в таблиці 6.4.

Таблиця 6.4 – Калькуляція кошторисної вартості робіт з розробки програми

Найменування статей витрат	Сума, грн.
Основна заробітна плата	25000
Додаткова заробітна плата	2500
ЄСВ	6050
Матеріали і комплектуючі	1050
Витрати на спеціальне обладнання	1436
Накладні витрати	12500
Разом	48536

6.3 Розрахунок економічної ефективності програмного продукту

Річну економію від впровадження програми, що дозволяє знизити витрати машинного часу ЕОМ для рішення певного завдання в порівнянні із програмою, що застосовувалася раніше, визначають за формулою (6.8):

$$E = (T_{м_1} - T_{м_2}) \cdot B_{екс.ктз} + E_н^a \cdot \frac{Ф_{б.ктз} \cdot (T_{м_1} - T_{м_2})}{Ф_{еф.ктз}} - \frac{S_{рп}}{T_c} + E_{зп}, \qquad (6.8)$$

де $T_{м_1}, T_{м_2}$ – машинний час, необхідний для розв'язання поставлених завдань, відповідно в старому та у новому варіантах, машино–год./рік;

$B_{екс}$ – експлуатаційні витрати, що доводяться на 1 год машинного часу ЕОМ, грн/ машино–год.;

$E_н^а$ – нормативний коефіцієнт ефективності капітальних вкладень у засоби автоматизації;

$Ф_{б.ктз}$ – балансова вартість ЕОМ (КТЗ), грн.;

$Ф_{еф.ктз}$ – річний ефективний фонд часу роботи ЕОМ, машино–год.;

$S_{рп}$ – сумарні витрати на розробку програми, грн.;

$T_с$ – термін служби впроваджуваної програми до її морального зношування, років. Приймаємо цей термін 5 років;

$E_{зп}$ – економія на заробітній платі.

До впровадження даного ПЗ дані функції виконувались інженером з аналізу даних 1800 годин на рік, а після 450 годин на рік (економія складає 0,75 ставки одного технолога з заробітною платою 10 тис. гривень на місяць).

Обчислимо:

$$E=(1800 – 450) \cdot 3{,}99 + 0{,}2 \cdot (20000 \cdot (1800 – 50)) / 1800 – 48536 / 5 +$$
$$+10000 \cdot 0{,}75 \cdot (1+0{,}22) \cdot 12 = 120479 \text{ грн.}$$

Коефіцієнт ефективності капітальних вкладень E и строк їхньої окупності $T_{ок}$ розраховуються за формулами (6.9) і (6.10):

$$E_ф = \frac{E}{S_{рп}}, \qquad (6.9)$$

$$T_{ок} = \frac{S_{рп}}{E}. \quad (6.10)$$

Обчислимо:

$$Еф = 120479 / 48536 = 2,48,$$

$$T_{ок} = 48536 / 120479 = 0,4 \text{ року}.$$

Зведемо отримані техніко-економічні показники в таблицю 6.5.

Таблиця 6.5 – Техніко-економічні показники

Показники	Сума	Одиниця виміру
Витрати на розробку програми	48608	грн.
Річна економія	120735	грн.
Строк окупності програми	0,4	рік

Таким чином, впровадження в експлуатацію розробленого програмного забезпечення на виробничому підприємстві дозволить скоротити 0,75 ставки одного інженера з аналізу даних з заробітною платою 10 тис. гривень на місяць. Економічні розрахунки показали, що строк окупності програми 0,4 року, що значно менше часу її нормативного строку окупності (5 років). Крім того, річна економія при використанні програми становить 120479 грн. Виходячи з виконаних розрахунків, можна зробити висновок, що розробка даного програмного продукту є економічно доцільною.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

Сукупність чинників виробничого середовища, що впливає на здоров'я і працездатність людини в процесі роботи, називається умовами праці. Організація і поліпшення умов праці на робочому місці є одним з важливих резервів продуктивності і ефективності праці.

В дипломної кваліфікаційної роботі проводиться дослідження та програмна реалізація багатопоточного алгоритму еволюційного пошуку.

При роботі фахівця постійно або періодично діють наступні небезпечні і шкідливі чинники:

– підвищені статичні навантаження користувачів ПК, пов’язані з повторенням однотипних рухів кистями рук може призвести до защемлення нерву у зап’ястному каналі (тунельний синдром);

– невідповідність ергономічних характеристик устаткування нормованим величинам внаслідок нераціональної організації робочого місця;

– можливість ураження електрострумом внаслідок небезпечного рівня напруги в електричному ланцюзі, тому у приміщені можуть виникнути аварійні ситуації: замикання, загоряння проводки, поразка електрострумом, що може привести до електротравм або летальних наслідків;

– невідповідність нормам параметрів мікроклімату;

– небезпеки, пов’язані з підвищеним рівнем шуму, який створюється устаткуванням та людьми, що знаходяться у приміщенні, і які призводить до погіршення слуху;

– небезпеки, пов’язані з підвищеним рівнем вібрації;

– незадовільні умови освітлення робочого місця, в наслідок не рівномірного розподілу освітлювальних приладів;

– можливість загорянь внаслідок порушень правил пожежної безпеки, що може призвести до пожеж;

– непоінформованість персоналу у відповідній послідовності дій в умовах надзвичайної ситуації, що призводить до травмування та до інших непередбачених негативних наслідків.

7.2 Заходи із забезпечення безпеки

Забезпечення безпеки праці, організації оптимальних умов праці, а також захисту навколишнього середовища досягається шляхом дотримання норм і проведення заходів, передбачених законодавчими актами України.

Для запобігання підвищеним статичним навантаженням, зокрема, «тунельному синдрому», користувачам ПК рекомендовано:

– для запобігання «тунельного синдрому» робоче місце має бути зручним і ергономічно продуманим;

– працюючи за клавіатурою, необхідно стежити, щоб кут згину руки в лікті був прямим, тобто 90 град;

– дотримуватися оптимальної висоти клавіатури від підлоги – 65-75 см;

– користуватися ергономічними і зручними особисто для вас миші і клавіатури;

– використовувати силіконові подушки в якості опори для кисті [20].

Згідно ПУЕ [21] приміщення виробництва відноситься до категорії «Приміщення без підвищеної небезпеки». При виконанні правил ПУЕ ураження електричним струмом можливо тільки у випадку несправності апаратури й живильних кабелів.

Апаратура, що перебуває під небезпечною для життя напругою, припускає забезпечення електробезпечності. Всі металеві частини заземлені на третій провід спеціальних триконтактних розеток. Струмопровідні кабелі розташовані під дерев'яною підлогою, зверху покритою лінолеумом, або у вигляді схованої проводки в стінах, таким чином, контакт із кабелем виключається. Над кожною розеткою розміщено попереджуючий надпис.

Електрична ізоляція забезпечується конструкціями пристроїв і способами їхнього підключення. У процесі експлуатації періодично проводиться перевірка якості ізоляції, шляхом виміру її опору [22]. Відповідно до ПУЕ в електроустановках до 1000 В мінімальне значення опору ізоляції силових і освітлювальних електропроводок на ділянці між суміжними запобіжниками або за останніми запобіжниками між будь-яким проведенням і землею становить не менш 0,5М Ом.

Відповідно до ПУЕ захисне заземлення застосовується у всіх електроустановках змінного струму під напругою 380В и вище й постійного струму 440В и вище. У приміщеннях з підвищеною небезпекою заземлення використовується в електроустановках змінного струму напругою 42 В та постійного струму 110 В [21]. Захисному заземленню підлягають корпуси трансформаторів, оболонки кабелів, металеві огородження електроустановок [22].

Вирівнювання потенціалів досягаються шляхом устрою контурних заземлювачів. У приміщеннях вирівнювання потенціалів здійснюється з'єднанням металевих стойок ЕОМ, корпусів апаратів світильників і іншого устаткування з усіма доступними для дотику металевими конструкціями будинку, спорудження, а також з контуром шиною захисного заземлення.

Електричний поділ мережі припускає розділ мережі на окремі електрично непов'язані між собою ділянки за допомогою розділових трансформаторів. Підвищення електробезпечності відбувається за рахунок зменшення ємкості мережі й підвищення опору ізоляції фаз мережі щодо землі.

Подвійна ізоляція забезпечує застосування крім основної, робочої ізоляції струмоведучих частин ще одного шару додаткової ізоляції, що ізолює людину від металевих неструмоведучих частин, які можуть випадково виявитися під напругою. При застосуванні електроустаткування з подвійною ізоляцією не потрібно ні заземлення, ні занулення їхніх корпусів [23].

Для зниження величини виникаючих зарядів статичної електрики покриття технологічних підлог варто виконувати з одношарового полівінілхлоридного антистатичного лінолеуму. Іншим методом захисту є нейтралізація заряду статичної електрики іонізованим газом. До загальних мір захисту від статичної електрики можна віднести загальне й місцеве зволоження повітря.

Існує небезпека попадання блискавок під час грози. Тому для попередження виникнення небезпечної ситуації за зазначених умов необхідно обладнати споруди, де знаходиться виробництво блискавковідводами (рис.7.1).

Визначимо, чи відповідає діючим вимогам блискавковідвід, розташований на даху спорудження, що відноситься до II категорії за рівнем блисковкозахисту.

Вихідні дані:

Довжина будинку, L=80 м;

Ширина будинку, S=40 м;

Найбільша висота будинку, h_x=10 м;

Фактична висота блискавковідводу, h=11 м;

Середня інтенсивність грозової діяльності – 40-60 годин на рік.

Середньорічна кількість ударів блискавки на 1 км^2 поверхні землі, n=4.

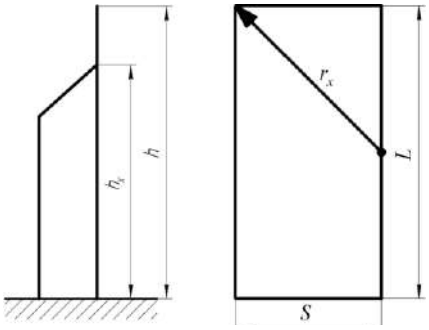


Рисунок 7.1 – Місцезнаходження блискавковідводу й розміри спорудження, що захищається

Категорія блискавкозахисту згідно ДНАОП 0.00-1.32-01 Правила устройства электроустановок. электрооборудование специальных установок - II [24].

Очікувана кількість ураження блискавкою будинку об'єкта без блискавкозахисту за рік розраховується на основі формули (7.1):

$$N = \left[\left(S + 6h_x\right)\cdot\left(L + 6h_x\right) - 7{,}7h_x^2\right]\cdot n\cdot 10^{-6}. \qquad (7.1)$$

Обчислимо:

$$N = \left[\left(40 + 6\cdot 10\right)\cdot\left(80 + 6\cdot 10\right) - 7{,}7\cdot 10^2\right]\cdot 4\cdot 10^{-6} = 0{,}053.$$

Т.к. N<1 – зона захисту приймається *Б*.

Зона захисту одинарного стрижневого блискавковідводу являє собою конус, тобто для захисту спорудження необхідно, щоб горизонтальний перетин зони захисту на висоті h, утворювало коло радіусом r_x. З рисунку 7.1:

$$r_x = \sqrt{S^2 + \left(\frac{L}{2}\right)^2} = \sqrt{40^2 + \left(\frac{80}{2}\right)^2} = 56{,}57 \text{ м.}$$

Приймаємо r_x = 56,57 м.

Необхідна висота блискавковідводу розраховується на основі формули (7.2):

$$\mathrm{h} = \frac{\left(\mathrm{r_x} + 1{,}63\mathrm{h_x}\right)}{1{,}5} = \frac{56{,}57 + 1{,}63\bullet 10}{1{,}5} = 48{,}52 \text{ м.} \qquad (7.2)$$

Приймаємо h = 49 м.

Оскільки фактична висота блискавковідводу в 4,5 рази менше розрахованої, блискавкозахист будинку виконаний невірно. Для того щоб блискавкозахист відповідав нормам необхідно встановити блискавковідвід висотою 49 м.

У розділі 4 ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [25] передбачені гігієнічні вимоги до організації устаткування робочих місць ВДТ ЕОМ і ПЭВМ. Конструкція робочого стола повинна відповідати сучасним вимогам ергономіки й забезпечувати оптимальне розміщення на робочій поверхні використовуваного устаткування (дисплея, клавіатури, принтера) і документів.

Висота робочої поверхні робочого стола із ВДТ повинні регулюватися в межах 680...800 мм, а ширина й глибина – забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600...1400 мм, глибина – 800..1000 мм). Приймаємо робочу поверхню з розмірами 1400х800. Щоб уникнути відблисків на екрані вікон доцільно розмістити робоче місце торцем до вікна не ближче чим на 1 м. Площа дозволяє нам розмістити додаткові меблі для зберігання документів, магнітних дисків, полками, стелажами, тумбами й т.п.

Велике значення надається ергономічним та естетичним вимогам. Цього домоглися за допомогою колірного рішення приміщення й устаткування з урахуванням вимог ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги» - у приміщеннях стіни пофарбовані фарбами пастельних тонів [26].

Під час організації робочого простору враховуються індивідуальні антропометричні параметри користувача з відповідними допусками на можливі зміни робочих поз та потребу у переміщеннях.

При організації роботи для збереження здоров'я, запобігання професійних захворювань і підтримки працездатності передбачені внутрізмінні регламентовані перерви для відпочинку. Тривалість обідньої

перерви визначається чинним законодавством про працю й Правилами внутрішнього трудового розпорядку на виробництві.

Для підвищення продуктивності праці, а також зниження рівня стомлюваності при роботі передбачені десятихвилинні перерви щогодини.

7.3 Заходи з виробничої санітарії та гігієни праці

Оптимальні норми температури повітря в робочій зоні виробничих приміщень у холодний і перехідний періоди року при II категорії робіт згідно ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» [27] – 20...23° С, відносна вологість – 60...40%, швидкість руху повітря – 0,2 м/сек. У теплий період року температура – 23...25° С, відносна вологість – 60...40%, швидкість руху повітря – 0,2 м/сек.

Для забезпечення вищевказаних параметрів передбачене застосування вентиляції - природної (аерації й провітрювання), механічної (загальнобмінної, місцевої приточної і витяжної), кондиціювання повітря.

У приміщенні забезпечений приплив свіжого повітря, кількість якого визначається техніко-економічним розрахунком і вибором схеми системи вентиляції. Мінімальна витрата повітря визначається з розрахунку 50...60 $м^3$/год на одну людину, але не менш дворазового повітрообміну в годину.

При невеликому забрудненні зовнішнього повітря кондиціювання приміщень здійснюється із перемінними витратами зовнішнього повітря й рециркуляційного.

Всі норми по освітленості беруться згідно ДБН В.2.5-28:2018 «Природне та штучне освітлення» [28]. Штучне освітлення в приміщеннях здійснюється системою загального рівномірного освітлення. При цьому штучне освітлення застосовують не тільки в темний, але й у світлий час доби.

Найбільш прийнятними для приміщень є люмінесцентні лампи ЛБ (білого світла) потужністю 20, 40 або 80 Вт, які встановлюються в

світильники типу ЛСП. Допускається застосування ламп накалювання у світильниках місцевого освітлення.

Загальне освітлення варто виконувати у виді суцільних чи переривчастих ліній світильників, розташованих збоку від робочих місць.

Згідно з ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» [29] зниження шуму, створюваного на робочих місцях внутрішніми джерелами, а також шуму проникаючого ззовні, є дуже важливим завданням. Зниження шуму в джерелі виникнення забезпечується застосуванням пружних прокладок між основою машини, приладу й опорною поверхнею. Як прокладки використовуються гума, повсть, пробка, різної конструкції амортизатори. Під настільні шумливі апарати можна підкладати м'які килимки із синтетичних матеріалів, а під ніжки столів, на яких вони встановлені, - прокладки з м'якої гуми, повсті, товщиною 6 - 8 мм.

Рівні вібрації під час виконання робіт з ЕОМ у виробничих приміщеннях не перевищують допустимих значень, визначених у ДСанПіН 3.3.2.007-98 «Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [25] та ДСН 3.3.6-039-99 «Державні санітарні норми виробничої загальної та локальної вібрації»

Нормування технологічної вібрації (загальної і локальної) відбувається:

– в залежності від її спрямування у кожній октавній смузі (1,6-1000 Гц);

– з середньоквадратичними віброшвидкостями (1,4-0,28)·10-2м/с;

– логарифмічними рівнями віброшвидкості (115-109 дБ);

– віброприскоренням (85-0,1) м/с2. [30].

7.4 Заходи безпеки у надзвичайних ситуаціях

7.4.1 Заходи з пожежної безпеки

В приміщенні ОЦ джерелом загоряння можуть бути електронні схеми ЕОМ, прилади для технічного обслуговування, електроустаткування, кондиціонери повітря, де за рахунок різних порушень відбувається перегрів тих чи інших елементів, електричні іскри, дуга, що можуть визвати загоряння горючих матеріалів, дерев’яні та пластикові стільці та столи, папір, перевантаження мережі живлення, людські чинники тощо.

Заходи щодо протипожежного захисту розділяються на організаційні, експлуатаційні, технічні й режимні (спеціальні).

Організаційні заходи: навчання робітників та службовців правилам пожежної безпеки, організація пожежної охорони, проведення бесід, лекцій, видання необхідних інструкцій, плакатів і т.п.

Технічні заходи передбачають дотримання протипожежних правил і норм при устрої систем опалення, вентиляції, кондиціюванні повітря, блисковкозахисті при спорудженні будинків, установці технологічного устаткування й ін.

Експлуатаційні заходи передбачають правильну експлуатацію систем опалення, вентиляції й кондиціювання повітря, блисковкозахисту технологічних машин і устаткування, правильне утримання будинків і територій і т.п.

Режимні заходи передбачають заборону або обмеження застосування відкритого вогню в вогненебезпечних місцях, паління в невстановлених місцях, обов'язкове дотримання норм і правил при роботі з вогненебезпечними й вибухонебезпечними речовинами [23].

Клас пожежі для приміщень ОЦ встановлюється згідно ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [31] – пожежі твердих речовин, переважно органічного походження, горіння

яких супроводжується тлінням (деревина, пластмаси, папір) – і визначається як клас А.

Категорія приміщення ОЦ за вибухопожежною небезпекою визначається згідно ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою» [32] – і визначається як пожежонебезпечність адмінприміщень.

Приміщення ОЦ обладнано автоматичними пожежними сповіщувачами, що реагують на підвищення температури, дим, полум'я, наприклад, сповіщувачі моделей ДТЛ, ІТМ.

Для відводу надлишкового тепла від ЕОМ задіяні кондиціювання повітря і система вентиляції, що обладнана пристроєм, який забезпечує автоматичне відключення вентиляції, в разі виникнення пожежі.

Напруга до електроустановок подається по кабельним мережам, що прокладені під технологічною з'ємною підлогою з рівнем вогнестійкості не менше 0,5 год.

До первинних засобів тушіння пожеж, призначених для ліквідації невеликих загорянь використовуються пожежні стволи, внутрішні пожежні водопроводи, вогнегасники. Згідно Правил експлуатації та типових норм належності вогнегасників [33] приміщення з ПЕОМ необхідно оснащувати переносними вуглекислотними вогнегасниками з розрахунку один вогнегасник ВВК-1,4 або ВВК-2 або ВВПА-400 на три ПЕОМ, але не менше ніж один вогнегасник, перелічених типів на приміщення.

У випадку пожежі передбачено шляхи евакуації робітників проходи, проїзди, евакуаційні виходи у відповідності до ДБН В.1.1.7-2002 «Пожежна безпека об'єктів будівництва» [31]. Евакуаційні виходи розташовано розосереджено у кількості не менше двох на споруду.

Коридори та проходи, що призначені для евакуації, мають як можна меншу довжину та мінімальну кількість поворотів. На усій протяжності проходу відсутні пороги або проміжні ступені.

Важливу роль у забезпеченні безпечного виходу людей грає протидимовий захист евакуаційних ходів. У будинках висотою до 9 поверхів незадимлюваність сходових кліток на час евакуації досягається їхньою ізоляцією від підвалів, горищ і поверхів. Для цього влаштовані відособлені входи в підвали, вхід на сходову клітку з поверхів здійснюють через тамбур - шлюз із підпором повітря, відокремлюють горища від сходових кліток перекриттями з негорючих матеріалів [23].

7.4.2 Заходи з цивільного захисту

Порядок дій сил цивільного захисту у вогнищі хімічного ураження.

Вогнища хімічного ураження (ВХУ) виникають у результаті застосування супротивником ОР, а також при руйнуванні хімічно небезпечних об'єктів, які виготовляють чи використовують у технології СДОР (такі вогнища прийнято називати вторинними). При надходженні перших даних про виникнення ВХУ начальники і штаби ЦО оповіщають населення в зоні зараження і у районах, яким загрожує небезпека від хмари зараженого повітря, яка поширюється, ставлять завдання хімічній і медичній розвідкам, віддають розпорядження про проведення заходів щодо захисту населення, приймають рішення й організовують проведення РІНР у вогнищах. До РІНР залучаються окремі батальйони спеціального захисту, підрозділи хімічного захисту військових частин ЦО, зведені загони (команди) протирадіаційного і протихімічного захисту, медичні формування, команди знезаражування й інші спеціально підготовлені й оснащені підрозділи і формування. собовий склад сил ЦО, що вводиться у ВХУ, забезпечується ЗІЗ органів дихання і шкіри, антидотами, індивідуальними протихімічними пакетами.

Першими у ВХУ для надання допомоги ураженим уводяться медичні підрозділи військових частин і формування медичної служби ЦО, а також підрозділи хімічного захисту та формування протирадіаційного і

протихімічного захисту. Основні завдання цих сил – надання негайної медичної допомоги ураженим, їх евакуація на незаражену місцевість і проведення дегазації території, споруд і техніки. У першу чергу евакуюють людей, які не мають засобів захисту органів дихання, потім тих, хто має протигази і вже одержали першу медичну допомогу; в останню чергу евакуюють людей, укритих в обладнаних фільтровентиляційними установками сховищах [34].

При проведенні рятувальних робіт у вторинному вогнищі ураження основні зусилля спрямовуються на локалізацію джерел отруйної речовини і запобігання її подальшого надходження на місцевість і в повітря.

Підрозділи хімічного захисту і формування протирадіаційного та протихімічного захисту в період проведення рятувальних робіт у ВХУ дегазують ділянки місцевості і доріг, будинки і споруди, проводять санітарну обробку особового складу військових частин, формувань і населення, знезаражують їх засоби захисту й одяг. Черговість проведення цих робіт визначається начальником ЦО, виходячи з конкретної обстановки. Для санітарної обробки населення, яке евакуюється з ВХУ, і дегазації транспортних засобів поблизу маршрутів евакуації на незараженій місцевості підрозділами хімічного захисту частин ЦО розгортаються пункти спеціальної обробки.

РІНР у ВХУ виконуються в протигазах і засобах захисту шкіри. Тривалість роботи змін у ВХУ залежить, головним чином, від припустимого часу безупинного перебування в ЗІЗ. Щоб уникнути перегріву і виходу його з ладу через теплові удари застосовують спеціальні комбінезони, що екранують, і поливання захисних костюмів водою.

Роботи проводяться з максимальною інтенсивністю до повного їх завершення в найкоротший термін, із залученням необхідної кількості сил і засобів та з дотриманням заходів безпеки. Заміна формувань здійснюється за рахунок резервів і залучення додаткових спеціальних формувань. У залежності від хімічної обстановки у вогнищах РІНР можуть проводитися

послідовно (спочатку в окремих, найбільш важливих місцях) або одночасне на всій території ВХУ. Вогнища вважаються ліквідованими, коли перебування людей без засобів захисту в них стає безпечним. На період проведення робіт вогнище ураження оточується, встановлюється комендантська служба й організовується охорона громадського порядку [35].

ВИСНОВКИ

У ході виконання дипломної кваліфікаційної роботи магістра було проведено огляд та аналіз методів еволюційної оптимізації. Відзначено, що еволюційні методи, як і інші методи оптимізації, можуть зациклюватися в областях локальних екстремумів і вимагають значних ресурсів комп'ютеру та часу для виконання оптимізаційного процесу.

Визначено, що в умовах обмеженості часу та при необхідності оптимізації складних нелінійних функцій доцільно використовувати паралельні методи еволюційного пошуку. Досліджено паралельні еволюційні методи. Відзначено, що при реалізації багатопоточної роботи еволюційних методів доцільно використовувати схему «майстер-робітник», оскільки така модель є досить простою для реалізації та більш ефективною у порівнянні з іншими моделями, особливо при роботі зі складними функціями.

Для програмної реалізації багатопоточного алгоритму еволюційного пошуку обрано мову програмування C#, оскільки швидкість виконання обчислень за допомогою цієї мови майже у два рази вище у порівнянні з іншими мовами. Крім того, мова C# ефективно підтримує реалізацію паралельних обчислень, що є важливим при програмуванні багатопоточних алгоритмів.

Розроблено структурну схему програми. Програма реалізована у вигляді клієнт-серверної моделі, на базі об'єктно-орієнтованого програмування. Клієнт надає користувачеві зручну можливість заповнити, форми для вводу даних, та надає доступ до результату роботи серверу. Має зручний та ергономічний інтерфейс з підказками для користувача та наданням користувачеві доступу до серверу з використанням мережевої технології, протоколу TCP. Підпрограма серверу реалізує основні функції системи та всі розрахунки.

Наукова новизна роботи полягає у тому, що набув подальшого розвитку багатопоточний алгоритм еволюційного пошуку на основі моделі

«майстер-робітник», в якому використовується подання цільових функцій на основі бінарних дерев, що дозволяє розпаралелити етап обчислення цільових функцій та суттєво пришвидшити процес еволюційної оптимізації.

Практичне значення роботи полягає у тому, що розроблено програмне забезпечення, що реалізує багатопоточний алгоритм еволюційного пошуку.

Виконано тестування розробленого програмного забезпечення. Виявлено, що запропонований багатопоточний алгоритм еволюційного пошуку дозволяє суттєво (в рази) прискорити процес еволюційної оптимізації у порівнянні з послідовними еволюційними методами. Графік прискорення свідчить про більш високу ефективність застосування розробленого багатопоточного алгоритму еволюційного пошуку у порівнянні з іншими паралельними реалізаціями еволюційних методів, зокрема методу на основі базової моделі «майстер – робітник» та острівної моделі еволюційного пошуку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Субботін С. О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей: монографія / С. О. Субботін, А. О. Олійник, О. О. Олійник ; під заг. ред. С.О. Субботіна. – Запоріжжя : ЗНТУ, 2009. – 375 с.

2. Олійник А. О. Еволюційні обчислення та програмування: Навчальний посібник / А. О. Олійник, С. О. Субботін, О. О. Олійник. – Запоріжжя : ЗНТУ, 2010. – 324 с.

3. Прогрессивные технологии моделирования, оптимизации и интеллектуальной автоматизации этапов жизненного цикла авиадвигателей : Монография / А. В. Богуслаев, Ал. А. Олейник, Ан. А. Олейник и др. ; под ред. Д. В. Павленко, С. А. Субботина. – Запорожье : ОАО «Мотор Сич», 2009. – 468 с.

4. Cantu-Paz E. Efficient and accurate parallel genetic algorithms / E. Cantu-Paz. – Massachusetts: Kluwer Academic Publishers, 2001. – 162 p.

5. Galichki M. Common Optimization of Adaptive Preprocessing Units and a Neural Network During the Learning Period / M. Galichki. – Application in EEG Pattern Recognition. Neural Networks. – 1997. – № 10. – P. 1153-1163.

6. Encyclopedia of artificial intelligence / Eds.: J. R. Dopico, J. D. de la Calle, A. P. Sierra. – New York : Information Science Reference, 2009. – Vol. 1-3. – 1677 p.

7. Gen M. Genetic algorithms and engineering design / M. Gen, R. Cheng. – New Jersey : John Wiley & Sons, 1997. – 352 p.

8. Haupt R. Practical genetic algorithms / R. Haupt, S. Haupt. – New Jersey : John Wiley & Sons, 2004. – 261 p.

9. Емельянов В. В. Теория и практика эволюционного моделирования / В. В. Емельянов, В. В. Курейчик, В. М. Курейчик. – М. : Физматлит, 2003. – 432 с.

10. Курейчик В. М. Генетические алгоритмы: монография / В. М. Курейчик. – Таганрог : ТРТУ, 1998. – 242 с.

11. Прогрессивные технологии моделирования, оптимизации и интеллектуальной автоматизации этапов жизненного цикла авиадвигателей : Монография / А. В. Богуслаев, Ал. А. Олейник, Ан. А. Олейник, Д. В. Павленко, С. А. Субботин ; под ред. Д. В. Павленко, С. А. Субботина. – Запорожье : ОАО «Мотор Сич», 2009. – 468 с.

12. Guliashki V. Survey of Evolutionary Algorithms Used in Multiobjective Optimization / V. Guliashki, H. Toshev, C. Korsemov. – Sofia: Institute of Information Technologies, 2009. – 13 p.

13. Рассел С. Искусственный интеллект: современный подход / С. Рассел, П. Норвиг. – М.: Вильямс, 2006. – 1408 с.

14. Ротштейн А. П. Интеллектуальные технологии идентификации: нечеткие множества, генетические алгоритмы, нейронные сети / А. П. Ротштейн. – Винница: Універсум-Вінниця, 1999. – 320 с.

15. Руденко О. Г. Штучні нейронні мережі / О. Г. Руденко, Є. В. Бодянський. – Х.: Компанія СМІТ, 2006. – 404 с.

16. Bentley J. L. Writing Efficient Programs (Prentice-Hall Software Series) [Text] / J. L. Bentley. – Englewood Cliffs, NJ : Prentice Hall, 1982. – 170 p.

17. Кнут Д. Искусство программирования [Текст] / Д. Кнут. – 3-е изд. – М.: Вильямс. — Т. 2, 2000 — 832 с.

18. C# язык программирования [Электрон. ресурс]. – 2019. – Режим доступа: https://ru.wikipedia.org/wiki/C_Sharp.

19. Экономика и организация производства программных продуктов [Электрон. ресурс]. – 2019. – Режим доступа: http://megaobzor.net/e-book/book.html

20. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. НПАОП 0.00-7.15-18. Державні санітарні правила і норми. / Харків: Форт, 2018 – 48 с.

21. Правила улаштування електроустановок [Текст] : ПУЕ-2017. – На заміну ПУЕ-86 ; чинний з 2017-08-21. – К. : Міненерговугілля України, 2017. – 617 с.

22. Жидецький В. Ц. Основи охорони праці [Текст] : підручник / В. Ц. Жидецький. – 5-те вид., доп. – К. : Знання, 2014. – 373 с.

23. Грибан В. Охорона праці / В. Грибан, А. Негодченко. К. : Центр навчальної літератури, 2017. – 280 с.

24. ДНАОП 0.00-1.32-01 Правила устройства электроустановок. электрооборудование специальных установок. Наказ Міністерства праці та соціальної політики України від 21.06.2001 г. № 272.

25. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Електрон. ресурс] : ДСанПіН 3.3.2.007-98. – Чинний від 1998-12-10. – К. : МОЗ України, 1998. – Режим доступу: http://mozdocs.kiev.ua/view.php?id=2445. – (Державні санітарні правила та норми).

26. ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги». Наказ від 21.12.2015 № 204 Про прийняття нормативних документів України, гармонізованих з міжнародними та європейськими нормативними документами, національних стандартів України та змін до національних стандартів України.

27. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. / Харків: Форт, 2000 – 24 с.

28. ДБН В.2.5-28:2018. Природне та штучне освітлення. Посібник: проблеми природного і штучного освітлення. / К: Етін, 2019. – 180 с.

29. ДСН 3.3.6.037-99. Санитарные нормы производственного шума, ультразвука и инфразвука. – К. : МОЗ Украины. – 1999. – 98 с.

30. Виробничий шум та вібрація // Довідник спеціаліста служби охорони праці. № 4, 2017. - С.34-54.

31. ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги». Київ : Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України, 2017. – 47 с.

32. ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою». Київ : Мінрегіонбуд України, 2016. – 66 с.

33. Правила експлуатації та типові норми належності вогнегасників. Київ, Офіційний вісник України від 30.03.2018 – 2018 р., № 25, стор. 124, стаття 913.

34. Шоботов В. М. Цивільна оборона [Текст] : Навчальний посібник / В. М. Шоботов. - Вид. 2-ге, перероб. – К. : Центр навчальної літератури, 2006. – 438 с.

35. Стеблюк М. І. Цивільна оборона та цивільний захист [Текст] : навч. посіб. для вузів / М. І. Стеблюк. – К. : Знання, 2013, – 487 с.

ДОДАТОК А

Технічне завдання

Вступ

Програма призначена для оброблення даних при оптимізації нелінійних багатовимірних функціональних залежностей.

А.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація багатопоточного алгоритму еволюційного пошуку», затверджене наказом по Національному університету «Запорізька політехніка».

А.2 Призначення розробки

Програма призначена для оптимізації нелінійних багатовимірних функціональних залежностей.

А.3 Вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Програма повинна використовувати архітектуру клієнт-сервер.

Клієнт має виконувати такі функції:

– надавати користувачеві можливість ввести усі необхідні для роботи дані;

– підтримувати зв'язок з сервером через протокол TCP;

– інтерфейс клієнту має бути зручний та інтуїтивно зрозумілий;

– реалізовувати вивід інформації, що надіслана сервером;

Сервер має використовувати наступні функції:

– реалізація ПГА;

– зв'язок з клієнтом у блокуючему режимі;

– синтаксичній аналіз формули, заданої строковим рядком та побудови бінарного дерева розрахунку.

А.3.2 Вимоги до надійності

Програма повинна забезпечувати коректну обробку помилок у випадку введення користувачем некоректних даних.

А.3.3 Вимоги до експлуатації

Для роботи з програмою користувач повинен володіти достатніми знаннями для коректної роботи з програмно-апаратним комплексом.

А.3.4 Вимоги до складу й параметрів технічних засобів

Для роботи з програмним продуктом рекомендується використовувати наступне апаратне забезпечення:

– ПК під керівництвом операційної системи;

– тактова частота процесора 2 ГГц або більше, кількість ядер – 2 або більше;

– обсяг оперативної пам'яті не менше 2 Гб;

– обсяг вільного простору на диску - не менше 100 Мб;

– дозвіл екрана монітора не менше 1024x768;

– маніпулятор миша.

А.3.5 Вимоги до інформаційної й програмної сумісності

Для роботи програми необхідна операційна система.

А.3.6 Вимоги до транспортування і зберігання

Програма повинна зберігатися на жорсткому або гнучкому магнітному диску і транспортуватися в спеціальних боксах.

А.3.7 Вимоги до програмної документації

Програма повинна поставлятися з технічним завданням, керівництвом оператора, керівництвом програміста.

А.4 Стадії та етапи розробки

Програмний продукт повинен бути виконаний у 16-ти тижневий строк з моменту отримання завдання.

ДОДАТОК Б

Опис програми

Б.1 Загальні відомості

Програма розроблена за допомогою мови програмування C#.

Б.2 Функціональне призначення

Програма призначена для оброблення даних при оптимізації нелінійних багатовимірних функціональних залежностей.

Б.3 Опис логічної структури

Програму розроблено на основі клієнт-серверної моделі, з використанням підходів об’єктно-орієнтованого програмування.

Клієнт надає користувачеві зручну можливість заповнити, форми для вводу даних, та надає доступ до результату роботи серверу. Має зручний та ергономічний інтерфейс з підказками для користувача та наданням користувачеві доступу до серверу з використанням мережевої технології, протоколу TCP.

Підпрограма серверу реалізує основні функції системи та все розрахунки. Для підвищення швидкості роботи сервер працює у блокуючому режимі, тобто одночасно може обробляти запит лише з одного клієнту, складається з таких модулів:

– модуль реалізації TCP з’єднання з клієнтом у блокуючому режимі;

– модуль синтаксичного аналізу формул, що аналізує символьній рядом та будує бінарне дерево за яким розраховується фітнес функції;

– модуль реалізації ПГА, що виконує основні функції програмами;

– модуль інтерфейсу що надає можливість адміністратору сервера, вибрати порт там запустити, або спинити сервер.

Б.4 Використані технічні засоби

Для роботи з програмою рекомендується використовувати наступне апаратне забезпечення:

– ПК під керівництвом операційної системи;

– тактова частота процесора 2 ГГц або більше, кількість ядер – 2 або більше;

– обсяг оперативної пам'яті не менше 2 Гб;

– обсяг вільного простору на диску - не менше 100 Мб;

– дозвіл екрана монітора не менше 1024x768;

– маніпулятор миша.

Б.5 Виклик і завантаження

Розроблене програмне забезпечення поширюється у вигляді інсталяційного пакета. Для можливості роботи з програмою необхідно розмістити файли модулів програмного забезпечення на сервері. Перед використанням програми її необхідно встановити. Для цього необхідно запустити програму setup.exe, яка знаходиться на установчому диску. У відповідь на запит установчої програми потрібно вказати шлях встановлення програми. Після цього програма скопіює необхідні файли, створить і помістить в головне меню системи ярлик для запуску програми і зробить необхідні зміни в файлі реєстрації. Після завершення процедури установки програму можна використовувати. Програму можна запустити будь-якими засобами операційної системи користувача. Для запуску програми необхідно запустити файл "ServApp.exe",”ClienApp.exe”.

Б.6 Вхідні дані

Вхідними даними для програми є характеристики, за якими буде проводитись розрахунок: розмір популяції, кількість поколінь (ітерацій розрахунків), кількість вимірів, цільова функція.

Б.7 Вихідні дані

Вихідними даними для програми є результати розрахунків (оптимальні значення цільової функції та значення вхідних параметрів, при яких досягаються відповідні значення цільової функції).

ДОДАТОК В

Текст програми

```
        //server
        using System;
        using System.Collections.Generic;
        using System.ComponentModel;
        using System.Data;
        using System.Drawing;
        using System.Linq;
        using System.Text;
        using System.Windows.Forms;
        using Kurs;
        using System.Threading;
        using System.IO;
        namespace ServerApp
        {


            public partial class Form1 : Form
            {
                [System.Runtime.InteropServices.DllImport("kernel32.dll")]
                extern static short QueryPerformanceCounter(ref long x);
                [System.Runtime.InteropServices.DllImport("kernel32.dll")]
                extern static short QueryPerformanceFrequency(ref long x);


                private delegate void getDataServerCallback(string buff);
                private Server serv = null;
                int _logFlag = 0;

                public Form1()
                {
                    InitializeComponent();
                }
                private void ServerReadi()
                {
                    StatusLb.Text = "server is ready to work";
                    RunBut.Enabled  = true;
                    StopBut.Enabled = false;
                }
                private void ServerNotReadi()
                {
                    StatusLb.Text = "server is not ready to work";
                    RunBut.Enabled = false;
                }
                private void Form1_Load(object sender, EventArgs e)
                {
                    this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.Fixed3D;
                    LogLb.BackColor        = Color.Transparent;
                    IpLb.BackColor         = Color.Transparent;
                    PortLb.BackColor       = Color.Transparent;
                    StatusLb.BackColor     = Color.Transparent;
                    LogFlag.BackColor      = Color.Transparent;
                    PortStatusLb.BackColor = Color.Transparent;
                    label1.BackColor            = Color.Transparent;

                    label1.Text =   "server busy";
                    label1.Visible = false;


                    String host = System.Net.Dns.GetHostName();
                    // Получение ip-адреса.
```

```
                    System.Net.IPAddress ip =
System.Net.Dns.GetHostByName(host).AddressList[0];
                    // Показ адреса в label'е.
                    IpLb.Text +="      "+ip.ToString();

                    PortStatusLb.Text = "Port is not entered";

                    LogFailEdit.Enabled = false;
                    StopBut.Enabled = false;

                    ServerNotReadi();

                }

                private void LogFlag_MouseClick(object sender, MouseEventArgs
e)
                {
                    if (_logFlag == 0)
                    {
                        _logFlag = 1;
                        LogFailEdit.Enabled = true;
                        LogFlag.Checked = true;
                    }
                    else
                    {
                        _logFlag = 0;
                        LogFailEdit.Enabled = false;
                        LogFlag.Checked = false;
                    }
                }

                private void maskedTextBox1_KeyPress(object sender,
KeyPressEventArgs e)
                {
                    if (e.KeyChar < 48 || e.KeyChar > 57)
                    {
                        if (e.KeyChar == 46 || e.KeyChar == 8)
                        {
                        }
                        else
                        {
                            e.KeyChar = Convert.ToChar(9);

                        }
                    }
                }

                private void textBox1_TextChanged(object sender, EventArgs e)
                {

                }

                private void PortEdit_TextChanged(object sender, EventArgs e)
                {
                    if (PortEdit.Text.Length > 0)
                    {
                        if (Convert.ToUInt32(PortEdit.Text) < 9999 &&
Convert.ToUInt32(PortEdit.Text) > 1)
                        {
                            PortStatusLb.Text = "";
                            ServerReadi();
```

```
                    }
                    else
                    {

                        PortStatusLb.Text = "port is incorrect";
                        ServerNotReadi();
                    }

                }
                else
                {
                    PortStatusLb.Text = "Port is not entered";
                    ServerNotReadi();
                }
            }

            private void RunBut_Click(object sender, EventArgs e)
            {
                serv = null;

                serv = new Server("127.0.0.1",
Convert.ToInt32(PortEdit.Text));
                serv.getData += new getDataHendler(main_getDataServer);
                RunBut.Enabled = false;
                StopBut.Enabled = true;
                StatusLb.Text = "server waits for clients";

            }

            public void main_getDataServer(object sender, string e)
            {
                this.Invoke(new getDataServerCallback(this.getDataServer),
new object[] { e });

            }
            private void getDataServer(string buff)
            {

                StatusLb.Text = "";
                label1.Visible = true;
                int varCount;
                string func;
                double maxValue, minValue;
                double multiMutation=0.2;
                int iterationCount;
                int indCount;
                double maxFitnes;
                int i=0;
                string locBuff="";

                while(buff[i]!=' ')
                {
                    if (buff[i] != '.')
                        locBuff += buff[i];
                    else locBuff += ',';
                    i++;
                }
                func = locBuff;
                locBuff = "";
                i++;
                while (buff[i] != ' ')
```

```
{
    if(buff[i]!='.')
    locBuff += buff[i];
    else locBuff += ',';
    i++;
}
maxValue = Convert.ToDouble( locBuff);
locBuff = "";
i++;
while (buff[i] != ' ')
{
    if (buff[i] != '.')
        locBuff += buff[i];
    else locBuff += ',';
    i++;
}
minValue = Convert.ToDouble(locBuff);
i++;
locBuff = "";
while (buff[i] != ' ')
{
    if (buff[i] != '.')
        locBuff += buff[i];
    else locBuff += ',';
    i++;
}
varCount = Convert.ToInt32( locBuff);
locBuff = "";
i++;
locBuff = "";
while (buff[i] != ' ')
{
    if (buff[i] != '.')
        locBuff += buff[i];
    else locBuff += ',';
    i++;
}
iterationCount = Convert.ToInt32(locBuff);
locBuff = "";

i++;
locBuff = "";
while (buff[i] != ' ')
{
    if (buff[i] != '.')
        locBuff += buff[i];
    else locBuff += ',';
    i++;
}
indCount = Convert.ToInt32(locBuff);
locBuff = "";

i++;
locBuff = "";
while (i< buff.Length)
{
    if (buff[i] != '.')
        locBuff += buff[i];
    else locBuff += ',';
    i++;
}
```

```
                    maxFitnes = Convert.ToDouble(locBuff);
                    locBuff = "";
                    double[] mas;
                    double fit;
                    Funk b = Funk.Instance;
                    b.loadFunk(varCount, func, maxValue, minValue,
multiMutation);

                    long ctr1 = 0;
                    long ctr2 = 0;
                    long freq = 0;
                    double time;
                    QueryPerformanceCounter(ref ctr1); // start
                    GA ga = new GA(iterationCount, maxFitnes,indCount, out
mas, out fit);
                    QueryPerformanceCounter(ref ctr2); // end
                    QueryPerformanceFrequency(ref freq);
                    time = (ctr2 - ctr1) * 1.0 / freq;


                    locBuff=" Fitnes = "+fit.ToString()+"\n Varible:\n";
                    for (i = 0; i < varCount; i++)
                    {
                        locBuff +=" "+mas[i].ToString() + "\n";

                    }
                    locBuff += " Time="+time.ToString();
                    serv.sendMsg(locBuff);
                    if (_logFlag == 1)
                    {
                        FileStream fs = new FileStream(LogFailEdit.Text,
FileMode.Append, FileAccess.Write);
                        StreamWriter sw = new StreamWriter(fs);
                        sw.WriteLine(buff);
                        sw.WriteLine(locBuff);
                        sw.Close();
                    }
                    Thread.Sleep(100);
                    serv.stop();
                    serv = null;
                    serv = new Server("127.0.0.1",
Convert.ToInt32(PortEdit.Text));
                    serv.getData += new getDataHendler(main_getDataServer);
                    RunBut.Enabled = false;
                    StopBut.Enabled = true;
                    StatusLb.Text = "server waits for clients";
                    label1.Visible = false;



                }

                private void StopBut_Click(object sender, EventArgs e)
                {
                    if (serv != null)
                    {
                        serv.stop();
                    }
                    timer1.Enabled = true;
                }
```

```
            private void Form1_FormClosing(object sender,
FormClosingEventArgs e)
            {
                if (serv != null)
                {
                    serv.stop();
                }
            }

            private void Form1_FormClosed(object sender,
FormClosedEventArgs e)
            {
                if (serv != null)
                {
                    serv.stop();
                }
            }

            private void timer1_Tick(object sender, EventArgs e)
            {
                ServerReadi();
                timer1.Enabled = false;
            }

        }
    }
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;

    namespace Kurs
    {

        class GA
        {
            int _generationCount; // количество генераций
            double _eps;          // порог
            int _individualCount; // колічесво особей в поуляции
            Funk _funk = Funk.Instance;  // получение екземпляра синлтона
формули
            Population _currentPop; // текушая популяция
            static Random rnd = new Random(System.Environment.TickCount);
            public GA(int generationCount, double eps, int
individualCount, out double[] args, out double result) // конструткор
запускающий генетыческий алгоритм
            {
                _generationCount = generationCount;
                _individualCount = individualCount;
                _eps = eps;
                int i=0;
                int flag = 1;
                double fitnes = start(); // первий шаг генрация начальной
виборки
                if (fitnes > eps)
                {
                    flag = 0;
                }
                while (flag == 1) // цикил в котором пока не будет
привішен лимит итераци й или достигунт нужній фитнес ведтся работа алгоритма
                {
```

```
                    fitnes = step(); // шаг алгоритма
                    if (fitnes > eps)
                    {
                        flag = 0;
                    }
                    if (i == generationCount)
                    {
                        flag = 0;
                    }
                    i++;

                }
                // вивод результата
                result = fitnes;
                args = _currentPop.getFirstAgs();
            }
            private double start()
            {
                _currentPop = new Population(_individualCount);
                _currentPop.newPopulation();
                _currentPop.sort();
                return _currentPop.getFitness();
            }
            private double step()
            {
                Population newPop = new Population(_currentPop.Count); //
создане новой популяцыи
                for (int j = 0; j < _currentPop.Count; j++) // выбор
особей для селекцыи
                {

                    int perent1 = rnd.Next(_currentPop.Count / 2,
_currentPop.Count - 1);
                    int perent2 = rnd.Next(_currentPop.Count / 2,
_currentPop.Count - 1);
                    int maxRand=0;
                    while (perent1 == perent2 || maxRand ==4)
                    {
                        perent2 = rnd.Next(_currentPop.Count / 2,
_currentPop.Count - 1);
                        maxRand++;
                    }
                    newPop.newPersonByIndex(j, _currentPop[perent1],
_currentPop[perent2]);// создания новой особи
                }
                newPop.sort(); // расчет фитнеса и сортировка по нему
популяцыи
                _currentPop = newPop; // присвоение текущей поуляцыи
значение новой
                return _currentPop.getFitness();
            }



        }
    }
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.Threading;
```

```
        namespace Kurs
        {
            class Person
            {
                double[] _params;
                Funk _funk = Funk.Instance;
                static Random _p;


                public Person()
                {
                    _p = new Random(DateTime.Now.Millisecond);
                    _params = new double[_funk.getParamCount()];
                }
                public void newPerson()
                {
                    for (int i = 0; i < _funk.getParamCount();i++)
                    {
                        _params[i] = ( _funk.getMaxValue() + 1 -
_funk.getMinValue()) * _p.NextDouble() + _funk.getMinValue();
                    }
                }
                public double[] getParams()
                {
                    return _params;
                }
                public double this[int x]
                {
                    get
                    {
                        if (x < _params.Length)
                        {
                            return _params[x];
                        }
                        else
                        {

                            return -1;
                        }
                    }
                    set
                    {
                        _params[x] = value;
                    }

                }
            }
                class Population
                {
                    Funk _funk = Funk.Instance;
                    int _personCount;
                    Person[] _population;
                    double[] _personFunckValue;
                    static Random _rnd = new
Random(System.Environment.TickCount);
                    static Random _p = new Random(DateTime.Now.Millisecond);

                    public Population(int personCount)
                    {
                        _personCount = personCount;
```

```
                    _population = new Person[_personCount];
                    _personFunckValue = new double[_personCount];
                    for (int i = 0; i < _personCount; i++)
                    {
                        _population[i] = new Person();
                    }

                }
                public void newPopulation()
                {
                    for (int i = 0; i < _personCount; i++)
                    {
                        _population[i].newPerson();
                    }
                }
                private void calculatePerson(int index)
                {

                    _personFunckValue[index] =
_funk.calculation(_population[index].getParams());
                }
                public void sort()
                {
                    Thread[] tmass = new Thread[_personCount];
                    for (int i = 0; i < _personCount; i++)
                    {
                        int buff = i;
                        tmass[buff] = new Thread(delegate() {
calculatePerson(buff); });
                        tmass[buff].Start();

                    }
                    for (int i = 0; i < _personCount; i++)
                    {
                        tmass[i].Join();
                    }
                    for (int i = 0; i < _personCount; i++)
                         for (int j = 0; j < _personCount-i-1; j++)
                            if (_personFunckValue[j] >
_personFunckValue[j + 1])
                            {
                                Person pbuff;
                                double dbuff;
                                pbuff = _population[j];
                                dbuff = _personFunckValue[j];
                                _population[j] = _population[j + 1];
                                _personFunckValue[j] =
_personFunckValue[j + 1];
                                _population[j + 1] = pbuff;
                                _personFunckValue[j + 1] = dbuff;
                            }


                }
                public void newPersonByIndex(int index, Person perent1,
Person perent2)
                {
                    Person chaild = new Person();
                    for (int i = 0; i < _funk.getParamCount(); i++)
                    {
                        chaild[i]=(perent1[i] +perent2[2])/2;
```

```
                    }

                    _population[index] = mutation(chaild);
                }
                private Person mutation(Person chaild)
                {
                    Person buff = chaild;
                    for (int i = 0; i < _funk.getParamCount(); i++)
                     if(_rnd.Next(0,2)==1)
                     {
                         buff[i] =chaild[i]*(1 + (_funk.multiMutation + 1
+ _funk.multiMutation) * _p.NextDouble() - _funk.multiMutation);

                     }
                    return buff;


                }
                public double getFitness()
                {
                    double buff = 0;
                    for (int i = 0; i < _personCount; i++)
                    {
                        buff += _personFunckValue[i];
                    }
                    return buff / _personCount;
                }
                public double getPersonValue(int i)
                {
                    return _personFunckValue[i];
                }
                public double[] getFirstAgs()
                {
                    double[] mas = new double[_funk.getParamCount()];
                    for(int i=0; i<_funk.getParamCount(); i++)
                    {
                       mas[i] =  this[_personCount-1][i];
                    }

                    return mas;
                }

                public Person this[int x]
                {
                    get
                    {
                        if (x < _population.Length)
                        {
                            return _population[x];
                        }
                        else
                        {

                            return _population[0];
                        }
                    }
                    set
                    {
                        _population[x] = value;
                    }
                }
```

```
                public int Count
                {
                    get
                    {
                        return _personCount;
                    }
                }
            }

                                        }


        using System;
        using System.Collections.Generic;
        using System.Linq;
        using System.Text;
        using System.Threading;

        namespace Kurs
        {
            class FuncGraf
            {
              int _znak;// 0 - "-", 1 - "+", 2-"/", 3 - "*"
              double _value;
              int _argIndex;
              int _type; //0 - value 1- arg
              static  string _error = "OK";
              static double[] _args;
              FuncGraf _a;  // левая ветвь дерва
              FuncGraf _b;  // правая ветвь дерева
              public FuncGraf(double value) //конструктор привнивающий ветве
знаения в виде числа
                {
                    _value = value;
                    _type = 0;
                    _error = "OK";

                }
              public FuncGraf(int index) // конструктор присваювающий елменту
значение індекса перемнной
                {
                    _argIndex = index - 49;
                    _type = 1;
                    _error = "OK";

                }

              public FuncGraf(int znak, FuncGraf a, FuncGraf b) // конструктор
присваевающий елменту значения виражения из правой и левой ветві
                {
                    _znak = _znak;
                    _a = a;
                    _b = b;
                    _type = 2;
                    _error = "OK";
                }
              public FuncGraf(string buff) // коструктор реалізующий первічний
ввод формули і аналіз віржения
                {
                    if (_error == "OK")
```

```
                {
                    _type = 2;
                    int i = 1;
                    string aBuff = "";
                    if (buff[0] == '(')
                    {
                        int count = 1;
                        i = 1;
                        while (count != 0)
                        {
                            if (count != 1 || buff[i] != ')')aBuff = aBuff +
buff[i];
                            if (buff[i] == '(') count++;
                            if (buff[i] == ')') count--;
                            i++;
                        }
                        _a = new FuncGraf(aBuff);
                    }
                    if (Char.IsDigit(buff[0]))
                    {
                        i = 0;

                        while (Char.IsDigit(buff[i]) || buff[i] == ',')
                        {
                            aBuff = aBuff + buff[i];
                            i++;
                        }
                        _a = new FuncGraf(Convert.ToDouble(aBuff));
                    }
                    if (buff[0] == 'x')
                    {
                        i = 0;
                       _a = new FuncGraf(Convert.ToInt32(buff[i+1]));
                       i+=2;
                    }
                    if (i < buff.Length)
                    {
                        switch (buff[i])
                        {
                            case '-': _znak = 0;
                                break;
                            case '+': _znak = 1;
                                break;
                            case '/': _znak = 2;
                                break;
                            case '*': _znak = 3;
                                break;
                            default: _error = "error in sintacsis";
                                break;

                        }
                        i++;
                        if (i < buff.Length)
                        {
                            string bBuff = "";
                            if (Char.IsDigit(buff[i]))
                            {

                                while (Char.IsDigit(buff[i]) || buff[i] ==
',')
                                {
```

```
                                    bBuff = bBuff + buff[i];
                                    i++;
                                    if (i == buff.Length) break;
                                }
                                if (i == buff.Length) _b = new
FuncGraf(Convert.ToDouble(bBuff));
                                else
                                {
                                    while (i != buff.Length)
                                    {
                                        bBuff = bBuff + buff[i];
                                        i++;
                                    }
                                    _b = new FuncGraf(bBuff);
                                }
                            }
                            else
                                if (buff[i] == '(')
                                {

                                    int count = 1;
                                    i++;
                                    while (count != 0)
                                    {
                                        if (count != 1 || buff[i] != ')')
bBuff = bBuff + buff[i] ;
                                        if (buff[i] == '(') count++;
                                        if (buff[i] == ')') count--;
                                        i++;
                                    }
                                    if (i == buff.Length)
                                    {
                                        _b = new FuncGraf(bBuff);
                                    }
                                    else
                                    {
                                        while (i != buff.Length)
                                        {
                                            bBuff = bBuff + buff[i];
                                            i++;
                                        }
                                        _b = new FuncGraf(bBuff);
                                    }

                                }
                                else
                                {
                                      if (buff[i] == 'x')
                                      {
                                        _b = new
FuncGraf(Convert.ToInt32(buff[i+1]));
                                      }
                                }

                        }

                    }
                }

            }
```

```
            public void setArgs(int count, double[] args) //функция задющия
аргументи
            {
                _args = new double[count];
                for (int i = 0; i <= count-1; i++)
                {
                    _args[i] = args[i];
                }
            }
            public void setArgs( double arg)
            {
                _args = new double[1];
                _args[0] = arg;

            }
            public void setArgs(double arg, double arg1)
            {
                _args = new double[2];
                _args[0] = arg;
                _args[1] = arg1;

            }
            public void setArgs(double arg, double arg1, double arg2)
            {
                _args = new double[3];
                _args[0] = arg;
                _args[1] = arg1;
                _args[2] = arg2;

            }
            public double calculate() //функция проізводяшая расчет
виражения по заднним аргументам
            {
                if (_error == "OK")
                {
                    if (_type == 0) return _value;  //возваршет  значения
числа елси данній елемнт дерва ветвь
                    else
                        if (_type == 1) return _args[_argIndex];
//возваршет  значения числа аргумент елси данній елемнт дерва ветвь
                    else
                    {
                        if (_b == null) return _a.calculate();
                        else
                        {
                            switch (_znak)  //реурсивній расчте текущего
елмента по левой и правой ветве и занку между ними
                            {
                                case 0: return _a.calculate() -
_b.calculate();

                                case 1: return _a.calculate() +
_b.calculate();
                                case 3: return _a.calculate() *
_b.calculate();
                                case 2:
                                    {
                                        double b = _b.calculate();
                                        if (b != 0)
                                        {
```

```
                                                return _a.calculate() /
_b.calculate();
                                            }
                                            else
                                            {
                                                _error = "error / by 0";
                                                return 0;
                                            }
                                        }
                                    }
                                }
                            }
                            else
                            {
                                return 0;
                            }
                            return 0;
                        }


        }
            class Funk
            {

                private sealed class FunkCreator
                {
                    private static readonly Funk instance = new Funk();
                    public static Funk Instance
                    {
                        get { return instance; }
                    }
                }
                public static Funk Instance
                {
                    get { return FunkCreator.Instance; }
                }


                int _paramCount;
                FuncGraf _funcGraf;
                double _maxParamValue;
                double _minParamValue;
                double _multiMutation;
                string _funcStr;

                protected  Funk() {}
                public void loadFunk(int paramCount, string func ,double
maxValue, double minValue, double multiMutation)
                {
                    _paramCount = paramCount;
                    _multiMutation = multiMutation;
                    _funcGraf = new FuncGraf(func);
                    _maxParamValue = maxValue;
                    _minParamValue = minValue;
                }



                public double calculation(double[] arg)
                {
```

```
                _funcGraf.setArgs(_paramCount, arg);
                return _funcGraf.calculate();
            }
            public int getParamCount()
            {
                return _paramCount;
            }
            public double getMaxValue()
            {
                return _maxParamValue;
            }
            public double getMinValue()
            {

                return _minParamValue;
            }
            public double multiMutation
            {
                get
                {
                    return _multiMutation;
                }
            }
        }

    }
    using System;
    using System.Collections.Generic;
    using System.Text;
    using System.Net;
    using System.Net.Sockets;
    using System.IO;
    using System.Threading;
    using System.Collections;
    using System.Windows.Forms;


    namespace ServerApp
    {
        public delegate void getDataHendler(object sender, string str);
//тип делегата для события получения сообщения
        class Server
        {

            public event getDataHendler getData; //создания события
            TcpClient tcpClient;                 //запускаем поток
прослушивания порта
            private StreamReader srReceiver;     //поток для отправления
данных(в строковом формате)
            private StreamWriter swSender;       //поток для получеия
данных (в строковом формате)
            private string strResponse;          //полученное сообщение
            private bool serverOn = false;       //флаг запуска  сервера
            private bool conetc = false;         //флаг установки
соеденения
            private TcpListener tlsClient;       //прослушиванеи порта
            Thread tLisen;                       //идентификатор для поток
для прослушки порта
            public Server(string IPadr, int port)
```

```
            {
                try
                {
                    tlsClient = new TcpListener(port); //инициализация
прослушки порта
                    tlsClient.Start();                 //запуск прослушки
порта
                    serverOn = true;                   //поднимаем флаг
запуска сервера
                    tLisen = new Thread(lisen);
//инициализация потока прослушки порта
                    tLisen.Start();                          //запуск
потока прослушки порта



                }
                catch
                {
                    serverOn = false;        //в случаее неудачи болка
трай опускаем флаг
                    MessageBox.Show("Не удалось запустить сервер, возможно
порт занят");
                }

            }
             private void lisen()
            {
                try
                {
                    tcpClient = tlsClient.AcceptTcpClient();  //запускаем
просшулку порта и получаем ссылку на клиент
                }
                catch
                {
                    tlsClient.Stop();
                }
                conetc = true;                              //поднимаем флаг
получения клиента
                srReceiver = new
System.IO.StreamReader(tcpClient.GetStream());  //перенаправляем поток чтения
                swSender = new
System.IO.StreamWriter(tcpClient.GetStream());    //перенаправляем поток
записи
                while ((strResponse = srReceiver.ReadLine()) != "" &&
serverOn)  //получаем сообщения
                {
                    // If it's invalid, remove the user
                    if (strResponse == null)  //если сообщения есть
                    {

                    }
                    else //если сообщения есть запускаем собітия получения
сообщения  иотдаем туда сообщения
                    {
                        // Otherwise send the message to all the other
users
                        onGetMsg(strResponse);
                    }
                }
```

```
                }
                 public void stop()  //отключаем сервер высвобождая все
использыванные классы(освобождаем порт)
                 {
                     if (serverOn)
                     {
                         tlsClient.Stop();

                         if (conetc)
                         {
                             tcpClient.Close();
                             conetc = false;
                             srReceiver.Close();
                             swSender.Close();
                         }
                         tLisen.Abort();


                     }
                 }
                 void onGetMsg(string buff)  //сообщения получения сообщения
                {
                  getData(null,buff);
                }
                 public void sendMsg(string buff)  //метод отправления
сообщения
                 {
                     swSender.WriteLine(buff);  //записываем в буфер
строку(буффер+символ конца строки)
                     swSender.Flush();          //особождаем поток в
канал(передаем данные в сесть)
                 }
            }
        }


        //client
        using System;
        using System.Collections.Generic;
        using System.ComponentModel;
        using System.Data;
        using System.Drawing;
        using System.Linq;
        using System.Text;
        using System.Windows.Forms;

        namespace ClientApp
        {
            public partial class Form1 : Form
            {
                private delegate void getDataClientCallback(string buff);
                private Client client = null;
                public Form1()
                {
                    InitializeComponent();
                }

                private void Form1_Load(object sender, EventArgs e)
                {
```

```
                this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.Fixed3D;
                panel1.BackColor = Color.Transparent;
                panel2.BackColor = Color.Transparent;
                panel3.BackColor = Color.Transparent;

                toolStripStatusLabel2.Text = "Ok";

            }


            private void getDataClient(string buff)
            {

                if (buff[0] != 'T') richTextBox1.Text += buff + "\n";
                    else            richTextBox1.Text += buff;

            }
            public void main_getDataClient(object sender, string e)
            {
                this.Invoke(new getDataClientCallback(this.getDataClient),
new object[] { e });
            }
            private void textBox5_MouseEnter(object sender, EventArgs e)
            {
             //  toolTip1.Show("Указывайте порядок вычислений явно, при
помощи скобок", textBox1);
            }

            private void textBox2_TextChanged(object sender, EventArgs e)
            {

            }

            private void textBox2_KeyPress(object sender,
KeyPressEventArgs e)
            {
                if (e.KeyChar < 48 || e.KeyChar > 57)
                {
                    if (e.KeyChar == 46 || e.KeyChar == 8 )
                    {
                    }
                    else
                    {
                        e.KeyChar = Convert.ToChar(9);

                    }
                }
            }

            private void textBox6_KeyPress(object sender,
KeyPressEventArgs e)
            {
                if (e.KeyChar < 48 || e.KeyChar > 57)
                {
                    if (e.KeyChar == 46 || e.KeyChar == 8)
                    {
                    }
                    else
                    {
                        e.KeyChar = Convert.ToChar(9);
```

```
                        }
                    }
                }

                private void textBox7_KeyPress(object sender,
KeyPressEventArgs e)
                {
                    if (e.KeyChar < 48 || e.KeyChar > 57)
                    {
                        if (e.KeyChar == 46 || e.KeyChar == 8)
                        {
                        }
                        else
                        {
                            e.KeyChar = Convert.ToChar(9);

                        }
                    }
                }

                private void textBox5_KeyPress(object sender,
KeyPressEventArgs e)
                {
                    if (e.KeyChar < 48 || e.KeyChar > 57)
                    {
                        if (e.KeyChar == 46 || e.KeyChar == 8)
                        {
                        }
                        else
                        {
                            e.KeyChar = Convert.ToChar(9);

                        }
                    }
                }

                private void IpEdit_KeyPress(object sender, KeyPressEventArgs
e)
                {
                    if (e.KeyChar < 48 || e.KeyChar > 57)
                    {
                        if (e.KeyChar == 46 || e.KeyChar == 8)
                        {
                        }
                        else
                        {
                            e.KeyChar = Convert.ToChar(9);

                        }
                    }
                }

                private void textBox3_KeyPress(object sender,
KeyPressEventArgs e)
                {
                    if (e.KeyChar < 48 || e.KeyChar > 57)
                    {
                        if (e.KeyChar == 46 || e.KeyChar == 8 || e.KeyChar ==
45)
                        {
```

```
                }
                else
                {
                    e.KeyChar = Convert.ToChar(9);

                }
            }
        }

        private void textBox4_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (e.KeyChar < 48 || e.KeyChar > 57)
            {
                if (e.KeyChar == 46 || e.KeyChar == 8 || e.KeyChar ==
45)
                {
                }
                else
                {
                    e.KeyChar = Convert.ToChar(9);

                }
            }
        }

        private void textBox8_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (e.KeyChar < 48 || e.KeyChar > 57)
            {
                if (e.KeyChar == 46 || e.KeyChar == 8)
                {
                }
                else
                {
                    e.KeyChar = Convert.ToChar(9);

                }
            }
        }

        private void IpEdit_MouseEnter(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "Enter IP adrres";
        }

        private void IpEdit_MouseLeave(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "";
        }

        private void PortEdit_MouseLeave(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "";
        }

        private void ExpressionEdit_MouseLeave(object sender,
EventArgs e)
        {
            toolStripStatusLabel2.Text = "";
```

```
        }

        private void textBox3_MouseLeave(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "";
        }

        private void textBox4_MouseLeave(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "";
        }

        private void textBox5_MouseLeave(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "";
        }

        private void textBox6_MouseLeave(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "";
        }

        private void textBox7_MouseLeave(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "";
        }

        private void textBox8_MouseLeave(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "";
        }

        private void PortEdit_MouseEnter(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "Enter pirt number";
        }

        private void ExpressionEdit_MouseEnter(object sender,
EventArgs e)
        {
            toolStripStatusLabel2.Text = "Enter expression";
        }

        private void textBox3_MouseEnter(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "Enter the maximum value of
the parameters";
        }

        private void textBox4_MouseEnter(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "Enter the minmum value of
the parameters";
        }

        private void textBox5_MouseEnter_1(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "Enter the numbers of the
parameters";
        }
```

```
        private void textBox6_TextChanged(object sender, EventArgs e)
        {

        }

        private void textBox7_MouseEnter(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "Enter the number of
individuals in the population";
        }

        private void textBox8_MouseEnter(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "Enter an adequate fitness";
        }

        private void toolStripStatusLabel2_Click(object sender,
EventArgs e)
        {

        }

        private void textBox6_MouseEnter(object sender, EventArgs e)
        {
            toolStripStatusLabel2.Text = "Enter the maximum number of
iterations";
        }

        private void button1_Click(object sender, EventArgs e)
        {

            if (client != null)
            {
                client.stop();
            }
            client = null;
            richTextBox1.Text = "";
            client = new Client(IpEdit.Text,
Convert.ToInt32(PortEdit.Text));
            client.getData += new
getDataClHendler(main_getDataClient);
            System.Threading.Thread.Sleep(10);
            client.sendMsg(ExpressionEdit.Text+" "+MaxValueEdit.Text+"
"+MinValueEdit.Text+" "+
                                NumberVarEdit.Text+" "+IterationEdit.Text+"
"+PopuletionEdit.Text+" "+
                                FitnesEdit.Text);

        }

        private void Form1_FormClosed(object sender,
FormClosedEventArgs e)
        {

            if (client != null)
            {
                client.stop();
            }
        }
```

```
            private void Form1_FormClosing(object sender,
FormClosingEventArgs e)
            {
                if (client != null)
                {
                    client.stop();
                }

            }



        }
    }

    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Text;
    using System.Windows.Forms;
    using System.Net;
    using System.Net.Sockets;
    using System.IO;
    using System.Threading;

    namespace ClientApp
    {
        public delegate void getDataClHendler(object sender, string str);
//тип делегата для собітия получения сообщения
        class Client
        {
            public event getDataClHendler getData;        //события
получения сообщения


            private StreamWriter swSender;               //поток для
записив клана
            private StreamReader srReceiver;             //поток для
чтения из канал
            private TcpClient tcpServer;                 //идентификатор
для тсп сервера
            private string strBuff;                      //полученное
сообщение
            Thread tLisen;
            private bool flagOn = true;                  //флаг
сведетльствующий о соеденении клиента с сервером
            public  Client(string IPadr, int port)
            {
                try                                                //блок
проверющий операции в случаее елси какято из операций пройдет неудачно будет
вызванно исключение
                {
                IPAddress ipAddr = IPAddress.Parse(IPadr);     //получаем
из конструктора IP адрес сервера
                tcpServer = new TcpClient();                   //выделяем
память под клас класс отвечающие за сервер в соеденении
                tcpServer.Connect(ipAddr, port);
//подключаемся к серверу
```

```
                    srReceiver = new
System.IO.StreamReader(tcpServer.GetStream());  //перенаправляем поток для
стения данных сервера
                    swSender = new
System.IO.StreamWriter(tcpServer.GetStream());    //перенаправляем поок для
отправки сообщений серверу
                    tLisen = new Thread(lisen);
//инициализаруем поток ожидания сообщения от сервера
                    tLisen.Start(); //запускаем поток ожидания сообщения от
сервера
                    }
                    catch
                    {
                        flagOn = false;
                        MessageBox.Show("Сервер не найден");

                    }


                }

                void lisen()
//метод юспользуемый в качесве поток для ожидания и получения сообщения от
сервера
                {
                    try
                    {
                        while ((strBuff = srReceiver.ReadLine()) != "" &&
flagOn == true)  //ждем сообщения от сервера если соеденение активно и сервер
не прислал пустую строку
                        {
                            if (strBuff == null)
                            {

                            }
                            else
                            {
                                onGetMsg(strBuff);  //вызываем события
получения сообщения
                            }

                        }
                    }
                    catch
                    {
                        stop();
                    }

                }
                void onGetMsg(string buff)  //описываем события получения
сообщения где выполняем функцию записанную в делегат
                {
                    getData(null,buff);
                }
                public void sendMsg(string buff) //отправка сообщения сервером
                {
                    if (flagOn == true)
                    {
                        swSender.WriteLine(buff);
                        swSender.Flush();
                    }
```

```
                }
                public void stop()          //особождения ресурсво в случаи
выключения клиента
                {
                    if (flagOn)
                    {
                        tcpServer.Close();
                        srReceiver.Close();
                        swSender.Close();
                        tLisen.Abort();
                        flagOn = false;
                    }
                }
            }
        }
```

ДОДАТОК Д

Слайди презентації

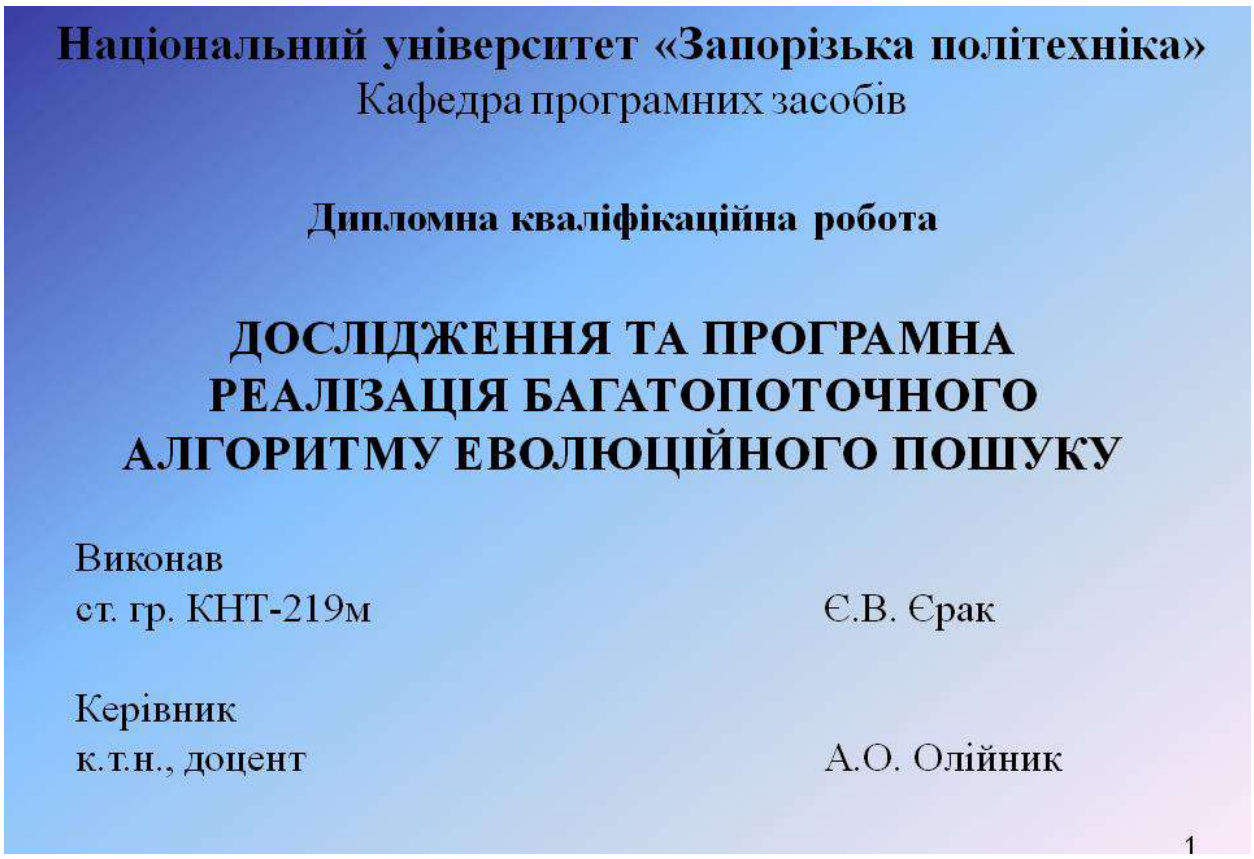


Рисунок Д.1 – Слайд № 1

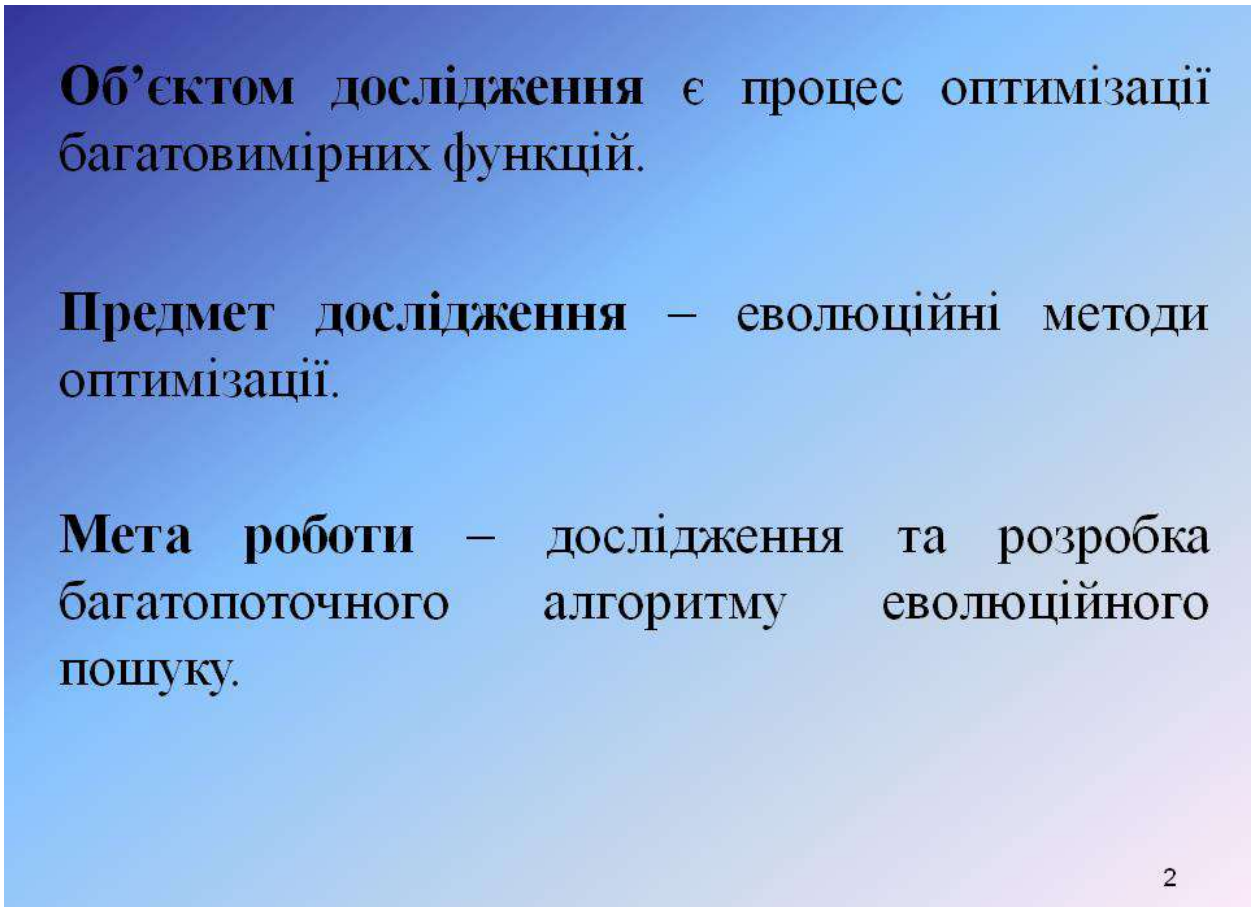


Рисунок Д.2 – Слайд № 2

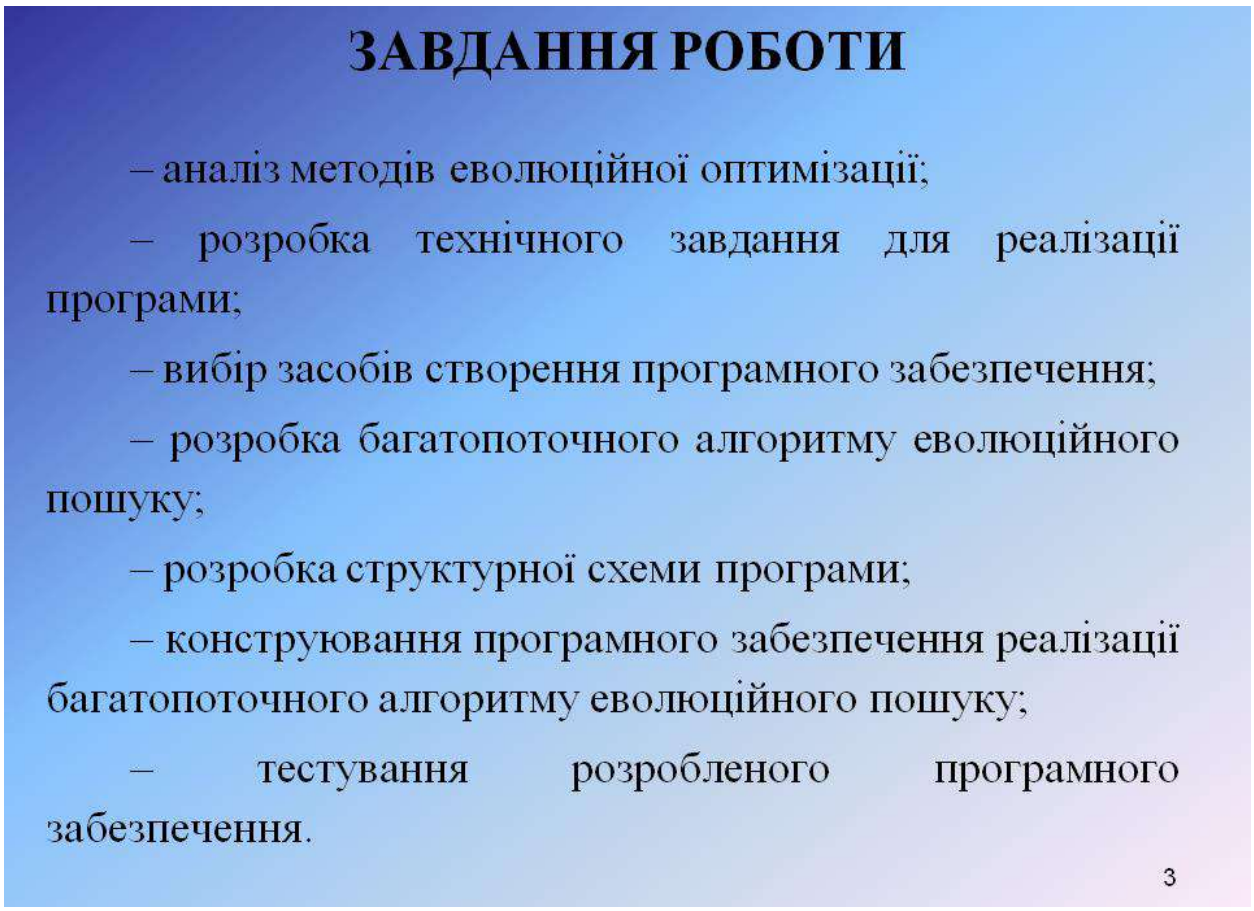


Рисунок Д.3 – Слайд № 3

Рисунок Д.4 – Слайд № 4

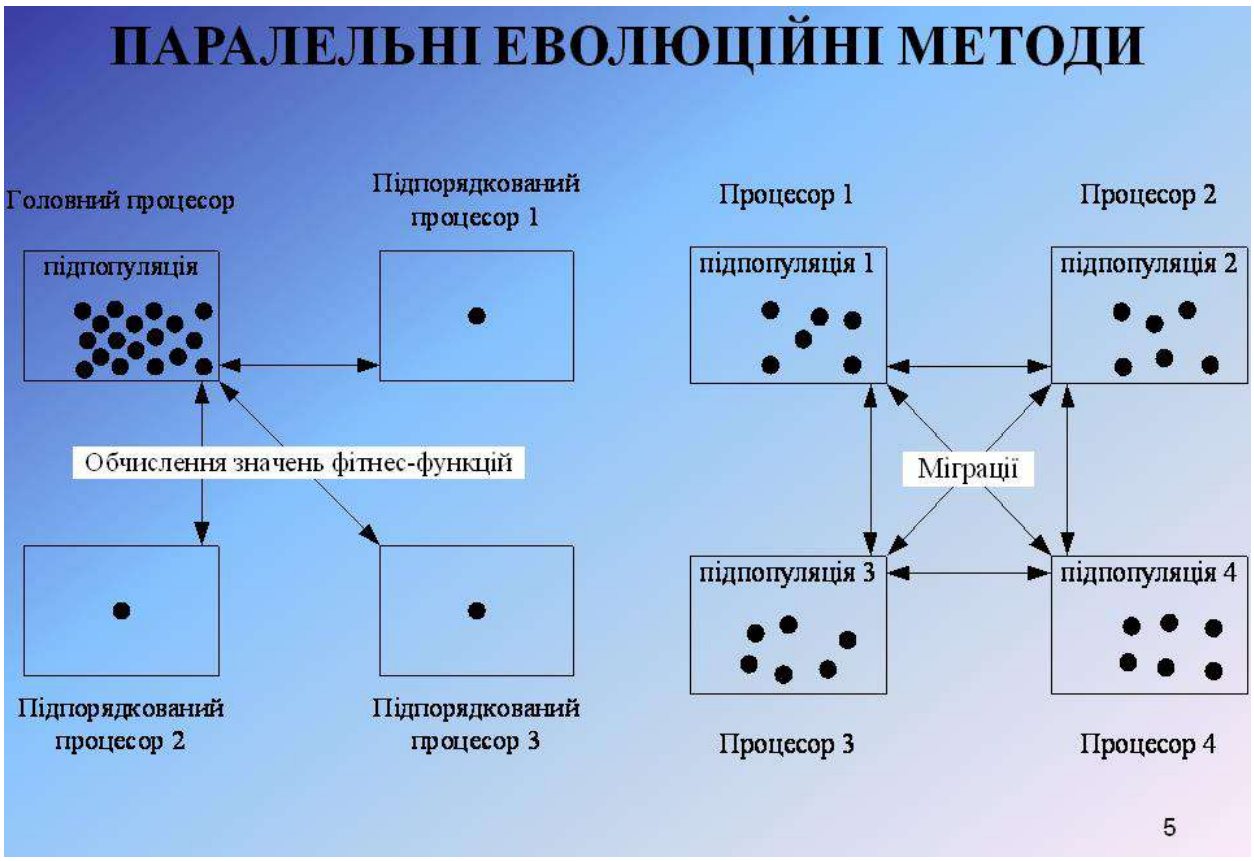


Рисунок Д.5 – Слайд № 5

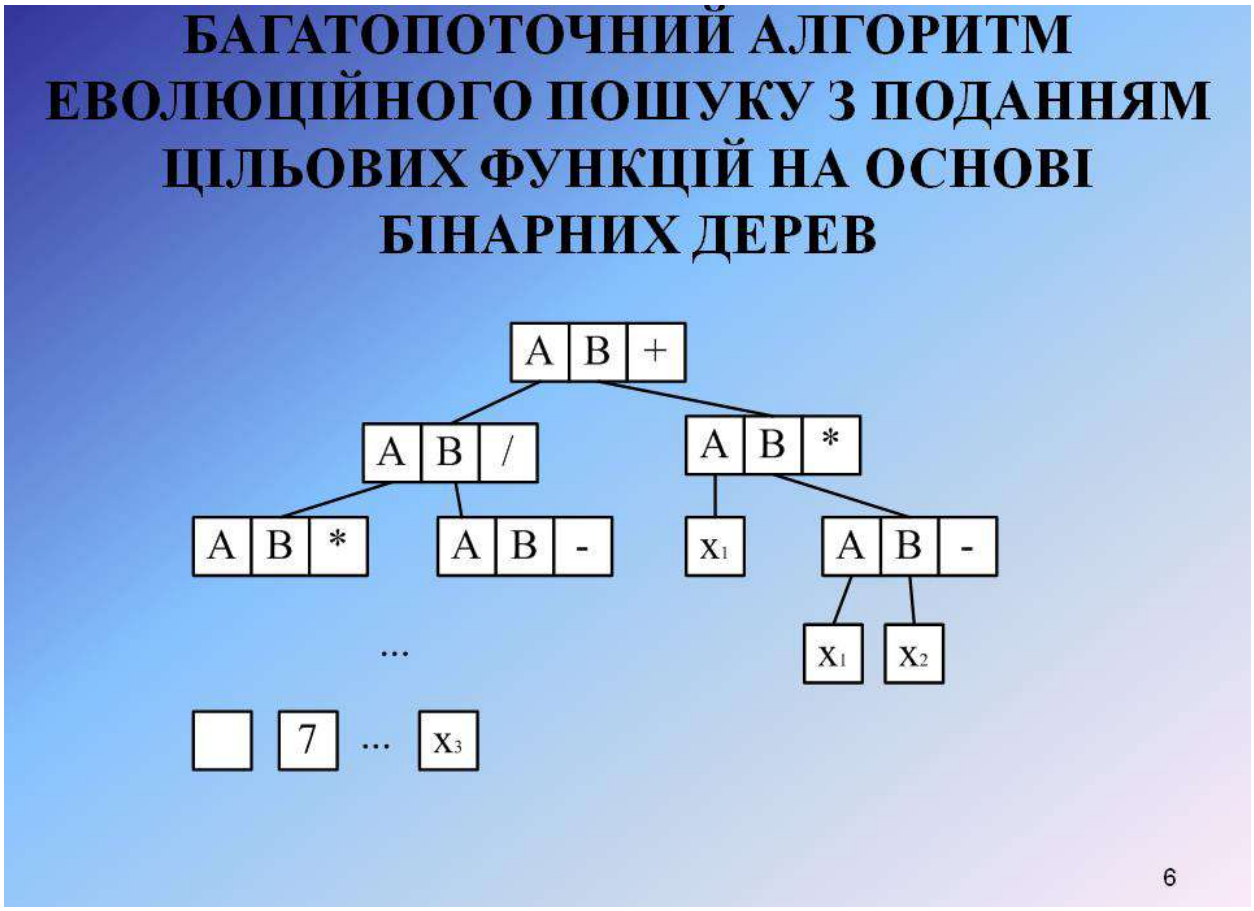
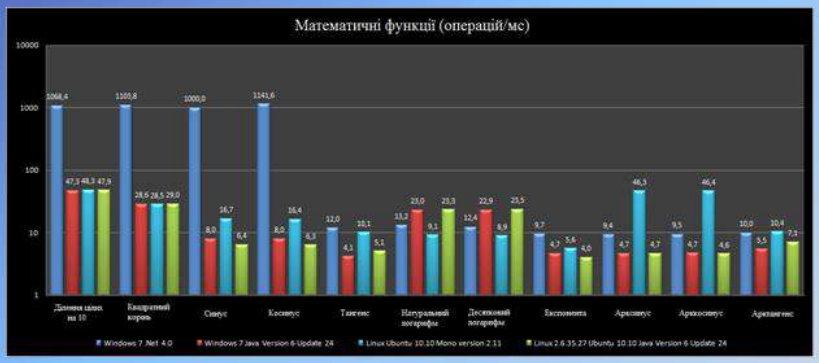
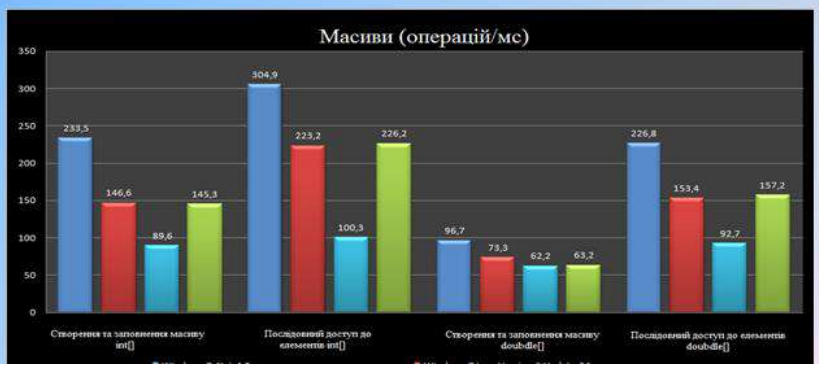


Рисунок Д.6 – Слайд № 6

ПОРІВНЯННЯ ШВИДКОСТІ РІЗНИХ МОВ ПРОГРАМУВАННЯ





7

Рисунок Д.7 – Слайд № 7

ПОРІВНЯЛЬНИЙ АНАЛІЗ МОВ ПРОГРАМУВАННЯ

Критерій порівняння	C#	Java	C++
Паралельні обчислення	+	+	±
Швидкість розробки	++	+	+
Розроблення графічного інтерфейсу користувача	+	+	+
Швидкість реалізації операцій на математичних функціях	±	+	++
Швидкість реалізації операцій при обробці масивів різних типів	–	+	++

8

Рисунок Д.8 – Слайд № 8

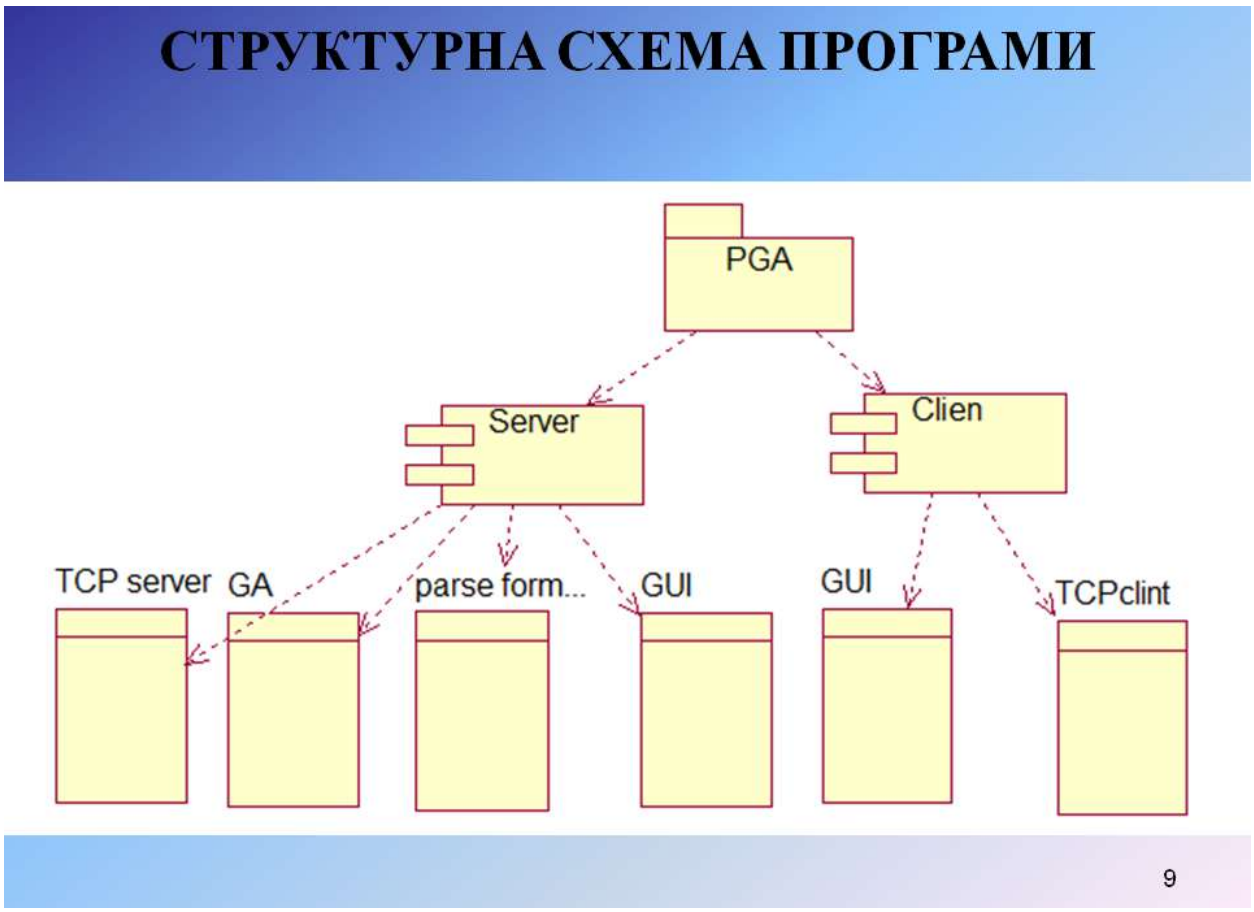


Рисунок Д.9 – Слайд № 9

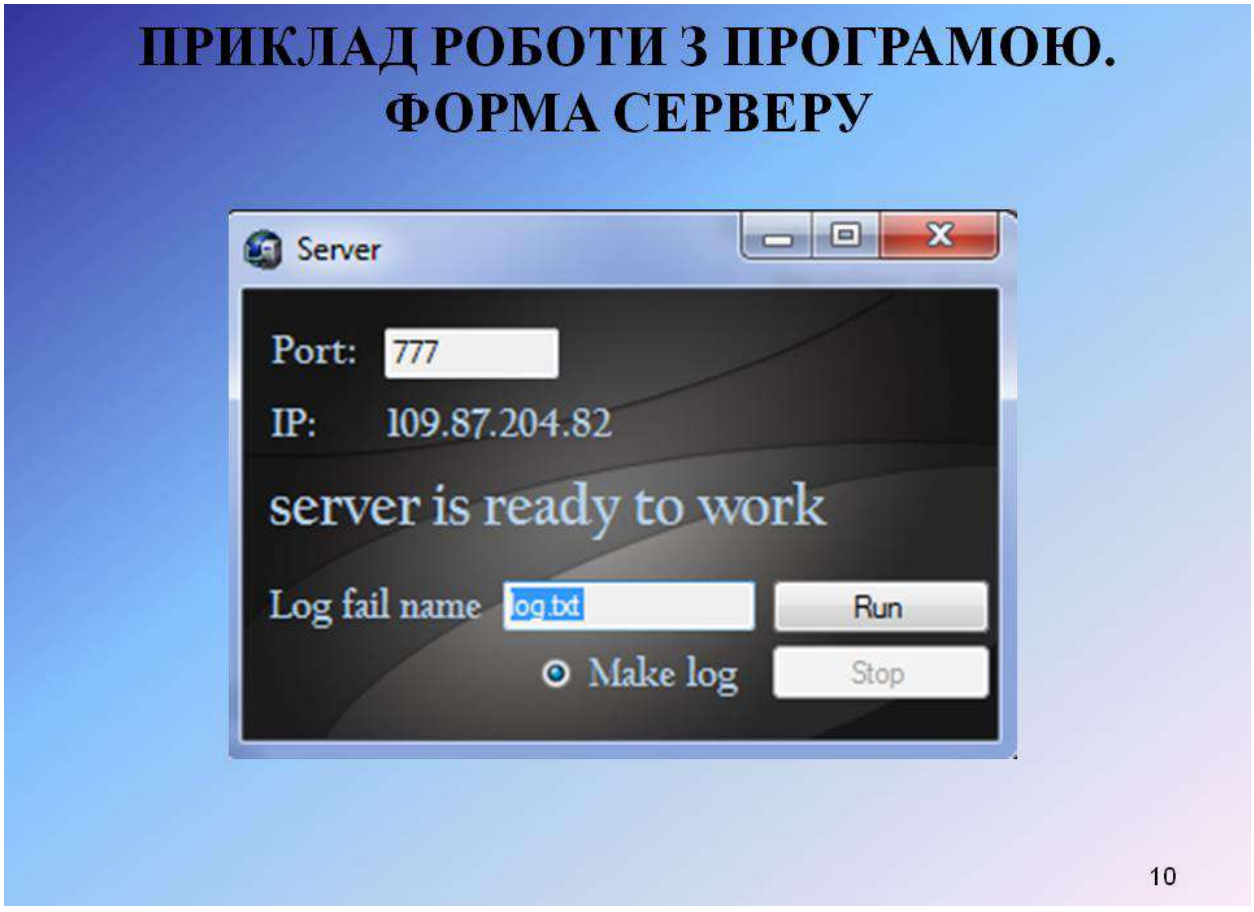


Рисунок Д.10 – Слайд № 10

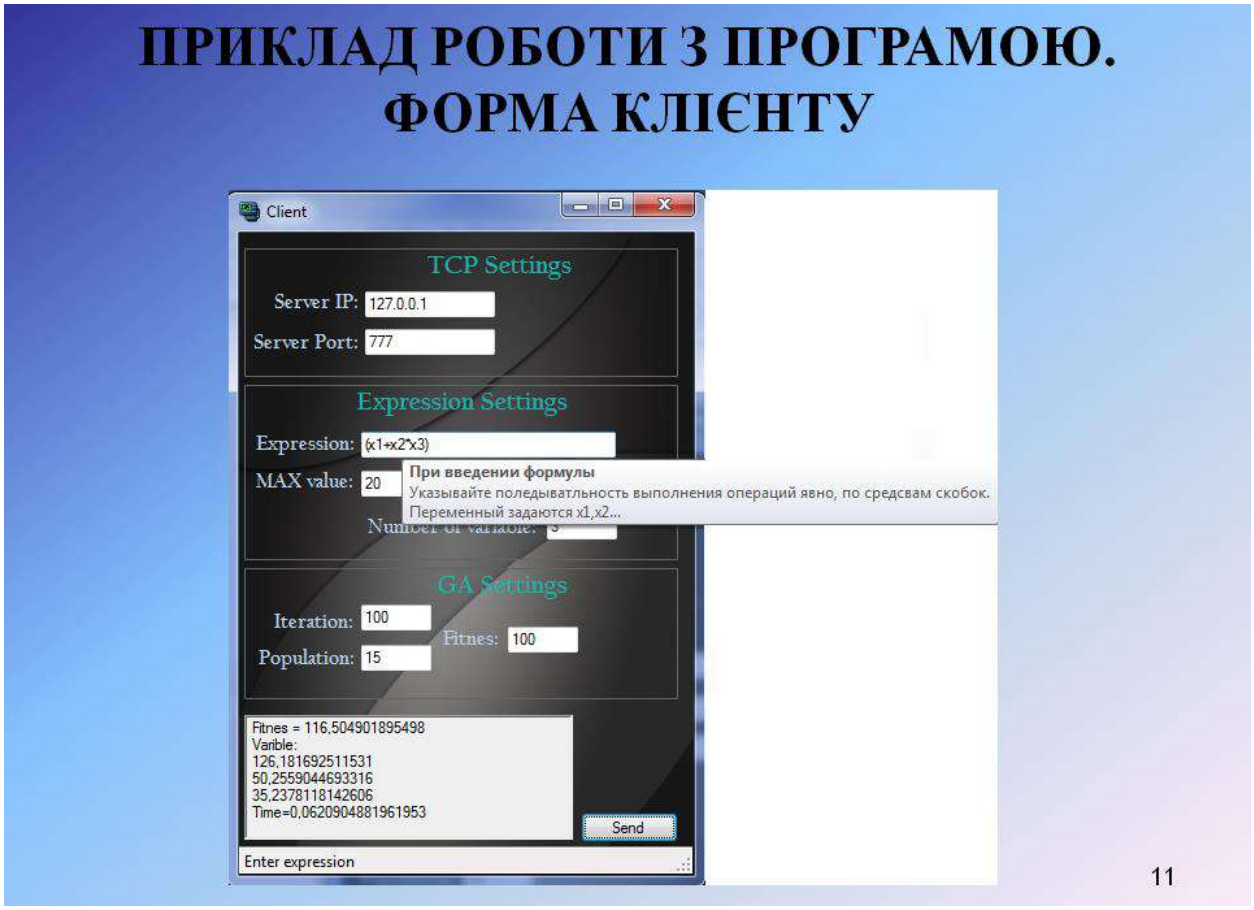


Рисунок Д.11 – Слайд № 11

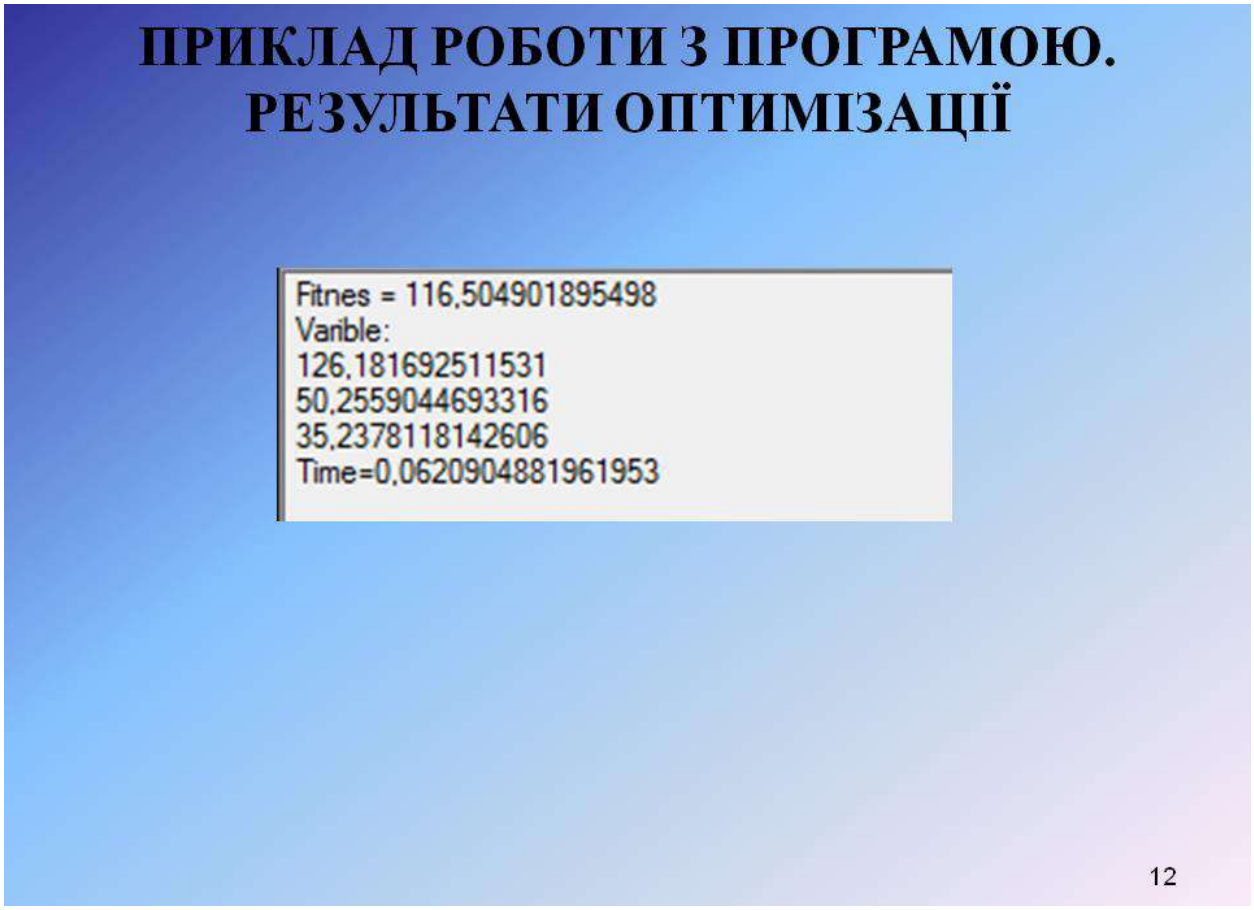
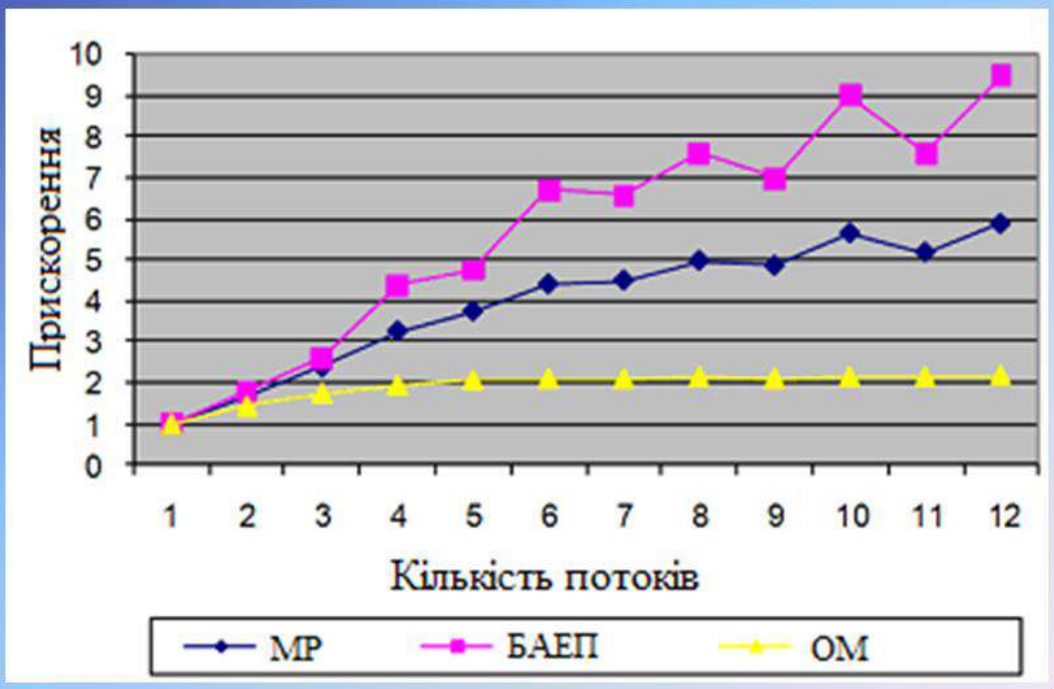


Рисунок Д.12 – Слайд № 12

ГРАФІК ПРИСКОРЕННЯ ПАРАЛЕЛЬНИХ ЕВОЛЮЦІЙНИХ МЕТОДІВ



13

Рисунок Д.13 – Слайд № 13

ВИСНОВКИ

В результаті виконання роботи розв'язано актуальне завдання дослідження та програмної реалізації багатопоточного алгоритму еволюційного пошуку.

Проведено огляд та аналіз методів еволюційної оптимізації. Відзначено, що еволюційні методи, як і інші методи оптимізації, можуть зациклюватися в областях локальних екстремумів і вимагають значних ресурсів комп'ютеру та часу для виконання оптимізаційного процесу.

Визначено, що в умовах обмеженості часу та при необхідності оптимізації складних нелінійних функцій доцільно використовувати паралельні методи еволюційного пошуку. Досліджено паралельні еволюційні методи. Відзначено, що при реалізації багатопоточної роботи еволюційних методів доцільно використовувати схему «майстер-робітник», оскільки така модель є досить простою для реалізації та більш ефективною у порівнянні з іншими моделями, особливо при роботі зі складними функціями.

14

Рисунок Д.14 – Слайд № 14

ВИСНОВКИ

Для програмної реалізації багатопоточного алгоритму еволюційного пошуку обрано мову програмування C#. Розроблено структурну схему програми. Програма реалізована у вигляді клієнт-серверної моделі, на базі об'єктно-орієнтованого програмування.

Наукова новизна роботи полягає у тому, що набув подальшого розвитку багатопоточний алгоритм еволюційного пошуку на основі моделі «майстер-робітник», в якому використовується подання цільових функцій на основі бінарних дерев, що дозволяє розпаралелити етап обчислення цільових функцій та суттєво пришвидшити процес еволюційної оптимізації..

Практичне значення роботи полягає у тому, що розроблено програмне забезпечення, що реалізує багатопоточний алгоритм еволюційного пошуку.

Виконано тестування розробленого програмного забезпечення. Виявлено, що запропонований багатопоточний алгоритм еволюційного пошуку дозволяє суттєво (в рази) прискорити процес еволюційної оптимізації у порівнянні з послідовними еволюційними методами.

15

Рисунок Д.15 – Слайд № 15