МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра_____Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти - другий (магістерський)

Дослідження моделі штучної імунної системи для вирішення задачі оптимізації виробництва
(тема)

Виконав: студент 2 курсу, групи ПЗСм-19-1___

_____Торба М.О._____
(прізвище, ініціали)

спеціальності 121- Інженерія програмного забезпечення
(код і повна назва спеціальності)

Освітньо-професійної програми
(тип програми)

Програмне забезпечення систем
(повна назва освітньої програми)

Керівник _____проф. каф. ПІ Четвериков Г.Г .
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2020 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет_Комп'ютерних наук_____

Кафедра_Програмної інженерії_____

Рівень вищої освіти - другий (магістерський)

Спеціальність _121-Інженерія програмного забезпечення_____

Тип програми _освітньо-професійна програма_____

Освітня програма_Програмне забезпечення систем_____

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Торбі Максиму Олеговичу_____

1. Тема роботи _Дослідження моделі штучної імунної системи для вирішення задачі оптимізації виробництва
затверджена наказом університету від "___" _____20 ___ р № _____
2. Термін подання студентом роботи до екзаменаційної комісії
16 грудня 2020 р.
3. Вихідні дані до роботи *теоретичні відомості про роботу біологічної імунної системи, алгоритм CLONALG для вирішення задачі оптимізації функцій кількох змінних. Використовувати ОС Windows, середовище розробки PyCharm*
4. Перелік питань, що потрібно опрацювати в роботі *аналіз предметної галузі, загальна характеристика біологічних і штучних імунних систем, опис алгоритму CLONALG для вирішення задачі оптимізації, постановка цілі і задачі дослідження, побудова математичної моделі виробництва, побудова математичної моделі штучної імунної системи, адаптація моделі імунної системи для вирішення задачі оптимізації побудованої моделі виробництва; розробка програмного додатку, аналіз впливу параметрів штучної імунної системи на збіжність алгоритму оптимізації і якість отримуваного оптимуму.*

5 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	Четвериков Г.Г.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Аналіз предметної галузі	06.11.2020	виконано
2.	Постановка задачі і цілі дослідження	10.11.2020	виконано
3.	Побудова математичної моделі виробництва і формулювання задачі оптимізації	13.11.2020	виконано
4.	Побудова математичної моделі штучної імунної системи	17.11.2020	виконано
5.	Створення програмного додатку	19.11.2020	виконано
6.	Проведення експерименту	24.11.2020	виконано
7.	Підготовка пояснювальної записки	27.11.2020	виконано
8.	Підготовка презентації та доповіді	02.12.2020	виконано
9.	Нормоконтроль, рецензування	09.12.2020	виконано
10.	Занесення диплома в електронний архів	10.12.2020	виконано
11	Попередній захист	11.12.2020	виконано
12.	Допуск до захисту у зав. кафедри	12.12.2020	виконано

Дата видачі завдання __02___ ____листопада____2020 р.

Студент _____
(підпис)

Керівник роботи _____ __проф. Четвериков Г.Г._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Атестаційна робота магістра містить 80 с., 11 рис., 11 табл., 9 формул, 14 джерел, 4 додатки.

АЛГОРИТМ CLONALG, АНТИТІЛА, ВИРОБНИЦТВО, КЛОНАЛЬНИЙ ВІДБІР, ОПТИМІЗАЦІЯ ВИРОБНИЦТВА, ОПТИМІЗАЦІЯ ФУНКЦІЇ КІЛЬКОХ ЗМІННИХ, ШТУЧНІ ІМУННІ СИСТЕМИ, PYTHON

Метою роботи є дослідження моделі штучної імунної системи для вирішення задачі оптимізація виробництва.

Методи розробки базуються на мові програмування Python і бібліотеці для побудови графіків Matplotlib.

В результаті роботи розглянуто існуючі моделі штучних імунних систем, побудовано математичні моделі виробництва та штучної імунної системи, сформулювано задачу оптимізації, розроблено програмний додаток для вирішення задачі оптимізації виробництва з використанням штучних імунних систем, досліджено вплив параметрів моделі на якість отриманих результатів.

ALGORITHM CLONALG, ANTIBODIES, MANUFACTURING, CLONAL SELECTION, MANUFACTURING OPTIMIZATION, MULTI-VARIABLE FUNCTION OPTIMIZATION, ARTIFICIAL IMMUNE SYSTEMS, PYTHON.

The purpose of the work is research on model of artificial immune system for solving manufacturing optimization problem.

Development methods are based on Python development tools and library Matplotlib for function plotting.

As a result of the work, was reviewed existing artificial immune system models, built models of factory and artificial immune systems, developed an application for solving the problem of factory optimization using artificial immune systems and investigated the influence of model parameters on the quality of results.

ПЕРЕЛІК СКОРОЧЕНЬ

Афінність	Міра міцності зв’язку між антитілом і антигеном. В термінології простору форм – відстань між ними
Збіжність алгоритму	Властивість ітераційних алгоритмів досягати оптимуму цільової функції або достатньо близько підходити до нього за скінченне число ітерацій
Простір форм	L-мірний простір, в якому вимірювання відповідають набору атрибутів, залучених до взаємодії антитіл і антигенів, що дозволяє представляти антитіла і антигени точками в цьому просторі. Як правило, антитіло і антиген мають одну й ту саму довжину L
ІС	Імунна система
ШІС	Штучна імунна система
CLONALG	Clonal Selection Algorithm

ЗМІСТ

ВСТУП 8
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ І ПОСТАНОВКА ЗАВДАННЯ 10
1.1 Загальна характеристика біологічної імунної системи 10
1.2 Загальна характеристика штучних імунних систем 11
1.3 Принцип клонального відбору. Алгоритм CLONALG 14
1.3.1 Теоретичні аспекти принципу клонального відбору 14
1.3.2 Алгоритм CLONALG для вирішення задачі оптимізації 16
1.4 Огляд літератури 18
1.5 Постановка цілі і задачі дослідження 21
2 МАТЕМАТИЧНА МОДЕЛЬ ВИРОБНИЦТВА 23
2.1 Опис виробництва 23
2.2 Функція цілі виробництва 24
2.3 Алгоритм роботи виробництва 28
3 МОДЕЛЬ ШТУЧНОЇ ІМУННОЇ СИСТЕМИ 31
4 ПРОГРАМНА РЕАЛІЗАЦІЯ 36
4.1 Загальний опис 36
4.2 UML-моделювання 39
4.3 Приклади коду і програмний вивод 41
5 ДОСЛІДЖЕННЯ ВПЛИВУ ПАРАМЕТРІВ МОДЕЛІ ШТУЧНОЇ ІМУННОЇ СИСТЕМИ НА РЕЗУЛЬТАТИ ОПТИМІЗАЦІЇ 44
5.1 Опис експерименту 44
5.2 Отримані результати 44
5.2.1 Вплив параметрів на процес оптимізації 44

5.2.2 Графіки значень цільової функції 47
5.3 Аналіз результатів експерименту 58
5.3.1 Вплив розміру популяції 58
5.3.2 Вплив кількості ітерацій 58
5.3.3 Вплив коефіцієнту клонування 59
5.3.4 Вплив зміни сторожового таймера 59
5.3.5 Вплив змін коефіцієнта обраних клітин 60
ВИСНОВКИ 61
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 62
ДОДАТОК А Перелік посилань на наукові дослідження університету 64
ДОДАТОК Б Апробація результатів роботи 65
ДОДАТОК В Слайди презентації 71
ДОДАТОК Г Електронні матеріали 80

ВСТУП

Протягом останніх двадцяти років спостерігається зростаючий інтерес до галузі штучних імунних систем (ШІС). Метою ШІС є використання ідей, отриманих з імунології, для розробки систем, здатних виконувати широке коло завдань у різних областях досліджень.

ШІС вивчені неглибоко, зокрема через невелику теоретичну базу. Їх застосування досліджено в багатьох, але не всіх сферах інженерної діяльності, і це дає можливості для пошуку і вирішення проблем за допомогою ШІС в нових і нових галузях. В даній роботі буде розглянуто, як ШІС вже використовуються в задачах розпізнавання символів, задачі комівояжера, планування роботи виробництва, (Job-Shop Scheduling Problem), оптимізації в електромагнетизмі, виявлення пошкоджень інструментів та інших.

Дана робота має на меті дослідити штучні імунні системи в задачі оптимізації виробництва, використовуючи принципи клонального відбору. Необхідно визначити, чи застосовні методи ШІС для вирішення задачі оптимізації виробництва, зокрема принципи клонального відбору.

Дослідження штучних імунних систем є однією зі сфер наукової діяльності кафедри Комп'ютерних інтелектуальних технологій та систем (КІТС) (проф. Корабльов М.М.) та кафедри електронно-обчислювальних машин (ЕОМ). На кафедрі Програмної інженерії дослідженнями в галузі моделювання та створення систем штучного інтелекту займається, зокрема, науково-навчальний дослідний центр "Математичне моделювання" (проф. Четвериков Г.Г.).

Мета дослідження – обґрунтувати застосовність ШІС в задачах оптимізації виробництва, перевірити збіжність алгоритму і визначити вплив параметрів моделі ШІС на результати оптимізації. Для досягнення мети роботи необхідно провести аналіз предметної області, побудувати математичну модель виробництва і поставити задачу оптимізації, побудувати математичну модель ШІС, в основі якої лежатимуть принципи клонального відбору, адаптувати модель для вирішення

задачі оптимізації змодельованого виробництва; розробити програмний додаток, який допоможе провести експеримент; інтерпретувати отримані результати.

Об’єкт дослідження – штучні імунні системи в задачах оптимізації. Предмет дослідження – побудована за принципами клонального відбору модель штучної імунної системи, яка вирішує задачу оптимізації змодельованого виробництва.

Побудовано модель виробництва і сформульовано задачу оптимізації. Експериментальним шляхом виявлено вплив окремих параметрів ШІС на результати оптимізації. Визначено вплив розміру популяції, кількості ітерацій, коефіцієнту відібраних клітин, коефіцієнту клонування та сторожового таймеру.

Для визначення впливу окремого параметру змінювали його значення, залишаючи інші параметри незмінними. Для кожної заданої конфігурації моделі обчислено найкращий розв’язок задачі за 100 запусків алгоритму. Побудувавши графіки цільової функції, зроблено висновки щодо впливу кожного параметра.

В результаті дослідження доведено застосовність ШІС в задачах оптимізації виробництва. Запропоновано удосконалений підхід до вибору клітин для заміни з обчисленням функції розподілу. Визначений вплив параметрів моделі на результати оптимізації. Проведені тести показують, що залежно від різних налаштувань параметрів алгоритм здатний знайти найкращий результат. Отримані значення цільової функції задовільні, проте через свій спосіб роботи алгоритм не завжди досягає глобального оптимуму в кожному запуску. Для малих значень кількості ітерацій та розміру популяції алгоритм є відносно ефективним з точки зору вартості обчислень, але іноді це обертається гіршою якістю отриманих результатів. З іншого боку, для вищих значень цих параметрів обчислювальна вартість більша, однак це призводить до отримання кращих результатів.

За результатами атестаційної роботи є публікація тез конференції. Також опубліковано тези, в яких описано використання ШІС в обчислювальних системах, зокрема розподілена класифікація з використанням мікросервісів і штучних імунних систем.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальна характеристика біологічної імунної системи

Перш ніж перейти до штучних імунних систем, варто розглянути, як працює справжня імунна система людини, дати основні визначення і зрозуміти, що надихнуло науковців до використання імунологічних принципів в обчислювальних системах.

Імунна система – це складна адаптивна структура, що захищає організм від патогенів [1]. Вона здатна розпізнавати величезну кількість молекулярних структур – антигенів, зокрема навіть тих, що не існують в природі і були синтезовані в лабораторії.

Захисні механізми імунітету поділяються на вроджені, які людина має від народження, та специфічні, яким організм навчається протягом життя.

Робота ІС забезпечується так званими імунокомпетентними клітинами – макрофагами та лімфоцитами, разом вони беруть участь в започаткуванні і розвитку усіх етапів специфічної імунної відповіді.

Лімфоцити зароджуються і розвиваються в первинних лімфоїдних органах. Поділяються на Т-лімфоцити і В-лімфоцити. Т-лімфоцити зароджуються в кістковому мозку, далі потрапляють в тимус і там досягають функціональної зрілості. В-лімфоцити утворюються і розвиваються повністю в кістковому мозку. Цікаво, що В-лімфоцити отримали свою назву по першій літері органу птахів – bursa fabricii, в якому були вперше виявлені.

При потраплянні в організм антигену клітини ІС намагаються його розпізнати [2]. Контакт лімфоциту з антигеном забезпечують вторинні лімфоїдні органи, завдання яких – «захопити» й утримати антиген. В ІС розпізнавання «свій-чужий» забезпечується Т-лімфоцитами й іншими клітинами, у яких на поверхні є рецептори, що здатні виявляти антигени. Взагалі, антиген – це будь-яка речовина, яку організм розглядає чужорідну або потенційно небезпечну і проти якої починає виробляти антитіла.

Розпізнавання антигенів стимулює процеси розмноження і диференціювання лімфоцитів. Вироблення антитіл – завдання саме В-лімфоцитів. Утворюються клони ідентичних антитіл, які нейтралізують антиген. Частина з утворених клітин зберігається для імунної пам'яті. В результаті подальший вплив антигену призводить до більш швидкої імунної реакції – вторинної відповіді Проте ресурси імунної пам’яті обмежені, тому вона постійно підсилює пам’ять до одного антигену і послаблює до іншого.

Різноманітність антитіл, що можуть бути вироблені, величезна - приблизно 100 мільйонів видів, тому наш організм може впоратися з більшістю патогенів.

Коротко, увесь процес виглядає наступним чином: патоген потрапляє всередину організму і його всіма способами атакує вроджена імунна система (фагоцити, лейкоцити, система комплементу). Інформація про патоген потрапляє у вторинні лімфоїдні органи (наприклад, в лімфовузли, за допомогою антигенпрезентуючих клітин), де з нею активно знайомляться клітини адаптивної імунної системи (Т-лімфоцити). Запускається виробництво антитіл за допомогою В-лімфоцитів. Антитіла приходять на допомогу і значно спрощують роботу вродженої імунної системи.

1.2 Загальна характеристика штучних імунних систем

В питаннях обробки даних корисними є наступні властивості ІС [3]:

− розпізнавання;

− різноманітність;

− пам’ять;

− розподілений пошук;

− ймовірнісне виявлення.

Так, основна задача, яку вирішує ІС, це визначення своїх і чужих клітин і вибіркової реакції на них.

ІС використовує комбінаторний механізм (генетично-обумовлений процес) для формування множини різних рецепторів лімфоцитів, для того щоб гарантувати, що хоча б один лімфоцит із всієї множини зможе взаємодіяти з будь-яким наперед заданим (відомим чи невідомим) антигеном.

Невелика кількість активованих лімфоцитів стають клітинами пам'яті (пам'ять ІС асоціативна). Використовуючи короткострокові і довгострокові механізми імунної пам'яті, ІС підтримує ідеальний баланс між економією ресурсів і виконанням функції за рахунок зберігання мінімально необхідної, але достатньої інформації про попередні контакти з антигеном.

Як було згадано раніше, ІС є розподіленою системою. Клітини ІС, головним чином лімфоцити, безперервно циркулюють по організму і в випадку зустрічі з антигеном виконують специфічну імунну відповідь.

Перехресні реакції в ході імунної відповіді – це стохастичний процес. До того ж виявлення антигену завжди неминуче відбувається приблизним чином; отже лімфоцит може взаємодіяти з кількома структурно схожими антигенами.

В імунній відповіді на антиген важливу роль грають й інші характеристики ІС: виділення особливостей, навчання, саморегуляція, пороговий механізм, спільна стимуляція, динамічний захист, здатність до адаптації, специфічність, самотолерантність та інші.

Усі ці властивості, які мають пряме відношення до обробки інформації, створюють ряд цікавих можливостей з точки зору обчислень. Відбувається розширення сфери застосування методів вирішення прикладних завдань, заснованих на принципах імунології. Ці методи можуть носити різні назви: штучні імунні системи (ШІС), системи, засновані на принципах імунітету, імунологічні обчислення тощо.

Штучні імунні системи – це адаптивні обчислювальні системи, побудовані на основі спостережуваних властивостей і принципів функціонування ІС і використовуються для вирішення завдань в різних галузях науки і техніки. Обчислювальні моделі ШІС включають в себе найбільш перспективні з точки зори обчислень властивості, які були описані вище.

На рисунку 1 показане місце штучних імунних систем в ієрархії штучного інтелекту.

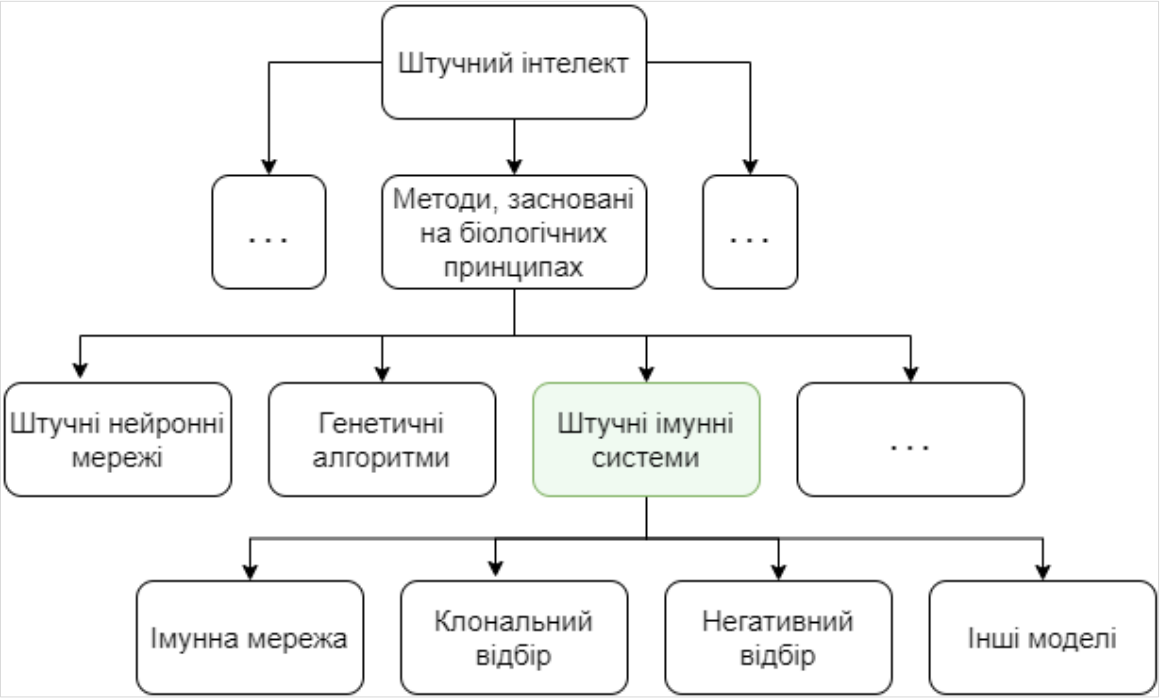


Рисунок 1 – Місце штучних імунних систем в ієрархії штучного інтелекту

Для пояснення механізмів імунітету існують різні теорії, є ряд імітаційних моделей, описуючих реакції різних компонентів імунного захисту. На сьогодні використовуються такі обчислювальні моделі:

− модель клонального відбору;

− модель імунної мережі;

− модель негативного відбору;

− модель позитивного відбору.

ШІС знайшли своє застосування в комп’ютерній безпеці: виявленні вірусів, моніторингу процесів в системі UNIX, задачах прогнозування [4], діагностиці несправностей; в задачах розпізнавання образів, сегментації аерофотознімків, задачах оптимізації і багатьох інших.

В ШІС використовується концепція імунних операторів, яка спрощує розуміння і реалізацію обчислювальних моделей ІС. Оператор ІС – це функція або

набор функцій, які являють собою один із етапів роботи ШІС. Далі описані деякі з них.

Оператор відбору призначений для відбору певної кількості антитіл з популяції для клонування і, як правило, заснований на афінності антитіл до антигенів. Кількість відібраних антитіл і механізм відбору задається конкретним алгоритмом.

Оператор клонування призначений для створення множини клонів антитіл. Клонування може бути статичним, коли кількість створюваних клонів не залежить від афінності даного антитіла до антигену; пропорційним, коли кількість створюваних клонів кожного антитіла пропорційна афінності даного антитіла к антигену, і обернено-пропорційна.

Оператор мутації призначений для внесення змін в рядок атрибутів, що представляє антитіло. Мутація має дві важливі функції: внесення різноманітності в популяцію антитіл і покращення афінності відібраних антитіл. За механізмами поділяється на статичну мутацію – ймовірність мутації фіксована і не залежить від афінності вихідного антитіла к антигену; пропорційну мутацію – ймовірність мутації пропорційна афінності антитіла к антигену, і обернено-пропорційну [5].

Оператор редагування популяції підтримує різноманітність популяції антитіл шляхом заміни антитіл з гіршою афінністю новими, згенерованими випадковим чином антитілами. Кількість антитіл, які підлягають заміні, задається конкретним алгоритмом.

1.3 Принцип клонального відбору. Алгоритм CLONALG

1.3.1 Теоретичні аспекти принципу клонального відбору

Принцип клонального відбору використовується для пояснення основних особливостей вторинної імунної відповіді на антиген. Згідно цього принципу для клонування відбираються ті клітини, які розпізнають лише антигени і не

розпізнають інші антитіла. Обрані клітини піддаються процесу мутації, або дозрівання, який покращує їх афінність із вибірковими антигенами. На рисунку 2 зображена схема процесу клонального відбору.

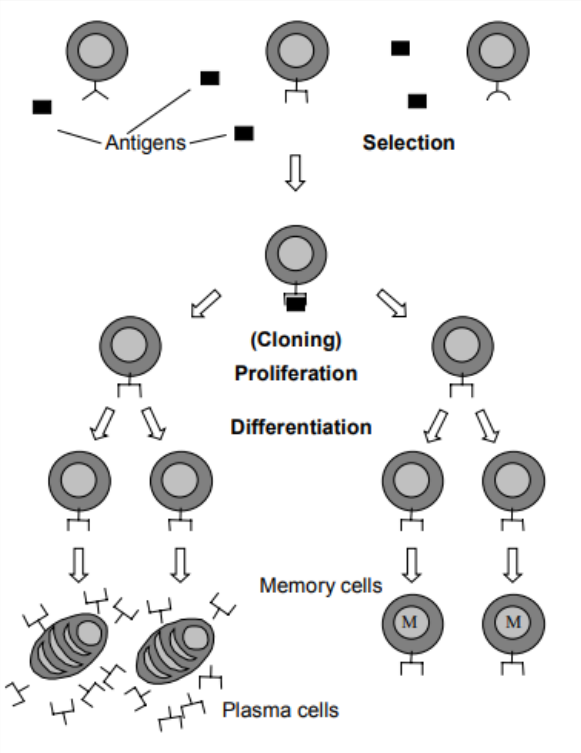


Рисунок 2 – Принцип клонального відбору

Основні кроки клонального відбору:

Крок 1 – представлення антигенів антитіл і обчислення афінності між ними.

Крок 2 – відбір антитіл з кращою афінністю.

Крок 3 – клонування відібраних антитіл.

Крок 4 – мутація клонів.

Крок 5 – представлення антигенів клонам і обчислення афінності між ними.

Крок 6 – відбір клонів з кращою афінністю і заміна ними вихідних антитіл.

Крок 7 – збереження клітин пам’яті для запам’ятовування знайдених рішень.

З точки зору обчислень важливі наступні аспекти теорії клонального відбору:

− розмноження та диференціація клітин по ступеню реакції на антиген;

− випадкова мутація клонів, що призводить до різноманітності популяції, цей процес називається дозріванням афінності;

− видалення тих нових лімфоцитів, які слабко реагують на антиген;

− збереження в пам’яті певного відсотка клітин з найкращою афінністю.

На обчислювальній моделі клонального відбору побудовано декілька алгоритмів. Найбільш популярні з них – алгоритм BCA (задачі оптимізації) і алгоритм CLONALG (оптимізація мультимодальних функцій, розпізнавання образів (бінарних зображень), комбінаторна оптимізація (задача комівояжера)) [6].

1.3.2 Алгоритм CLONALG для вирішення задачі оптимізації

Алгоритм CLONALG, спочатку розроблений для вирішення задачі розпізнавання образів, був адаптований для вирішення задачі оптимізації мультимодальних функцій. Алгоритм CLONALG для вирішення задачі оптимізації представляє таку послідовність кроків [7].

Крок 1 – генерація популяції антитіл Ab з випадковими характеристиками. Популяція антитіл – безліч можливих рішень задачі оптимізації. Кожне антитіло являє собою точку в багатовимірному просторі форм, яка є можливим оптимумом.

Крок 2 – обчислення афінності антитіл популяції Ab до антигену. В якості антигену виступає цільова функція. В даному випадку афінність кожного антитіла буде обчислюватись як значення цільової функції в точці, координати якої закодовані в даному антитілі. Для обчислення афінності значення координат точки підставляються в функцію цілі.

Крок 3 – клонувати всі антитіла популяції n разів. При цьому створюється тимчасова популяція клонів.

Крок 4 – мутація клонів. При цьому створюється популяція C^*.

Крок 5 – обчислення афінності клонів популяції C^*.

Крок 6 – відбір клонів і заміна ними відповідних антитіл. З популяції C^* обираються антитіла з покращеною в результаті мутації афінністю, ними замінюються відповідні антитіла, від яких були зроблені клони.

Крок 7 – редагування популяції. Замінити d антитіл з гіршою афінністю новими випадково згенерованими антитілами для підтримки різноманітності популяції. Чим гірша афінність антитіл, тим більше шансів, що вони будуть замінені новими.

Крок 8 – перевірка критерію зупинення. Якщо критерій зупинення (задана кількість поколінь, приріст функції мети або ін.) досягнуто, то результатом оптимізації буде антитіло з кращою (максимальною) щодо популяції афінністю. В іншому випадку повернутися до кроку 2.

На рисунку 3 зображено схему роботи алгоритма CLONALG для вирішення задачі оптимізації мультимодальних функцій.

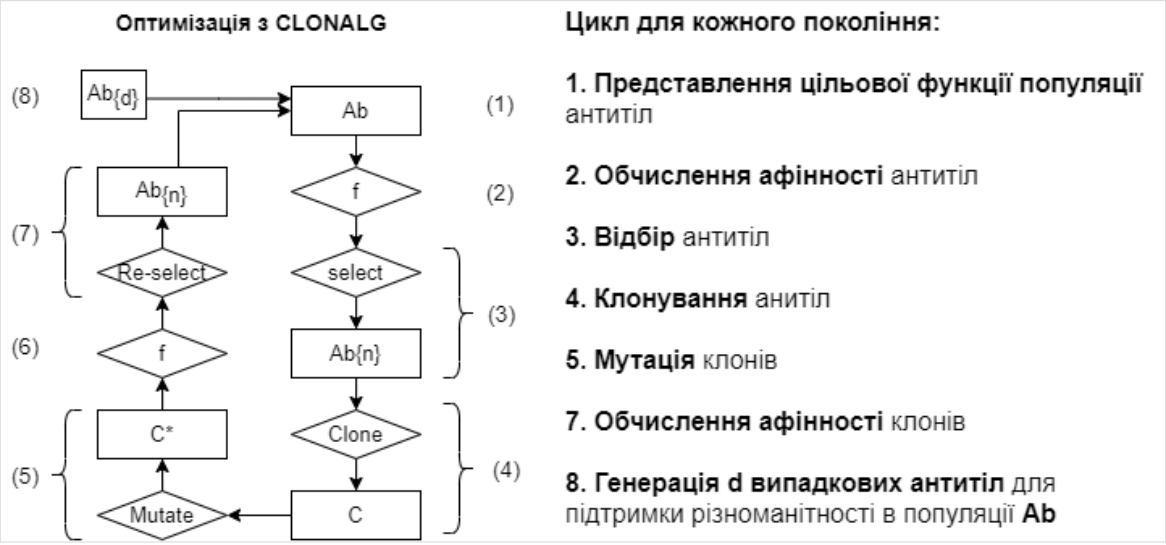


Рисунок 3 – Схема алгоритму CLONALG для вирішення задачі оптимізації мультимодальних функцій

На кожному поколінні роботи алгоритму відбувається покращення афінності антитіл, в результаті чого антитіла переміщуються по поверхні функції в бік її зростання. Результатом роботи алгоритму буде одне антитіло з популяції, для якого значення цільової функції буде максимальним. Тому немає необхідності домагатися максимальної афінності для всіх антитіл популяції. Однак з кожним поколінням роботи алгоритму середня по популяції афінність буде зростати. На рисунку 4 показаний розподіл антитіл на поверхні функції з ростом числа

проведених ітерацій: зліва – розподіл вихідної популяції, посередині – розподіл на середніх ітераціях, справа – розподіл на останніх ітераціях.

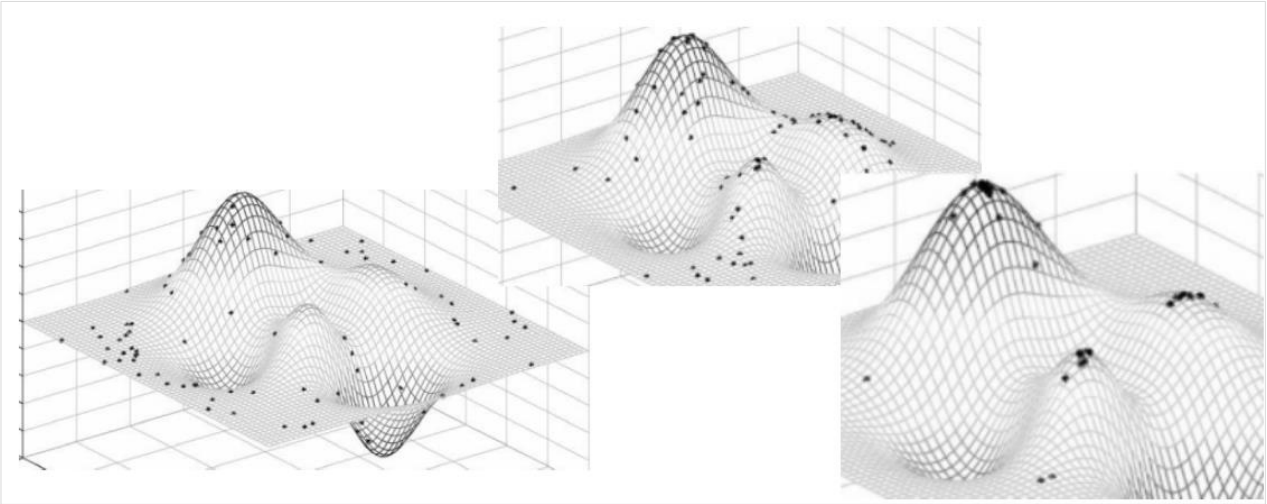


Рисунок 4 – Переміщення антитіл по поверхні функції з кожними наступними ітераціями

Максимум функції може бути досягнутий значно раніше, ніж закінчаться усі ітерації. Більш того, зі збільшенням числа ітерацій необов’язково усі антитіла прагнуть до точки глобального максимуму, як видно на рисунку 4 справа. При певних налаштуваннях алгоритму частина антитіл може «застрягти» в точках локальних максимумів і вибратись з них не зможе. Параметри алгоритму впливають на швидкість збіжності [8] та обчислювальну складність алгоритму.

1.4 Огляд літератури

У статті [9] розглядається використання алгоритму клонального відбору для оптимізації в електромагнетизмі. Запропоновано нову реалізацію імунного алгоритму – RCSA. Отримані результати роботи алгоритму є перспективними, оскільки RCSA може досягати високих коефіцієнтів збіжності при низькій кількості обчислень цільових функцій. RCSA здатний знайти не лише глобальний

оптимум, але і набір локальних оптимумів, що дає розробнику можливість обрати найбільш підходяще рішення з деяких високопродуктивних конфігурацій.

RCSA також успішно оптимізував пристрій SMES за прийнятих обчислювальних витратах. Знайдено рішення, порівнянне з потужним генетичним алгоритмом та найкращим рішенням, доступним у літературі, при цьому не виходячи за обмеження. Ці результати показують доцільність RCSA у вирішенні реальних проблем електромагнітної оптимізації.

У статті [3] представлено використання ШІС в промислових цілях. Виходячи з параметрів механічної обробки (зусилля, крутний момент тощо) та збурень (вібрацій тощо), автори виявляють пошкодження інструменту. В випадку промислового застосування, «свої» клітини – це звичайні операції різання, а «чужі» – це будь-яке відхилення, що перевищує допустимі зміни сили різання. Результати показують, що з використанням алгоритму негативного відбору поломка інструменту була виявлена у всіх тестових випадках.

В [10] порівняно алгоритми штучного інтелекту з алгоритмом клонування та механізмами соціального навчання. У цій статті пропонується алгоритм ШІС з механізмами соціального навчання (AIS-SL) для складних задач оптимізації. Враховуючи, що просте елітарне навчання (термін алгоритмів навчання) має великий ризик потрапити в оптимуми, пропонований AIS-SL розглядає два механізми соціального навчання: стохастичне соціальне навчання (SSL) та евристичне соціальне навчання (HSL). Крім того, пропонується стратегія динамічного оновлення радіуса пошуку для підвищення точності рішення. В числовому моделюванні продуктивність запропонованого AIS-SL порівнюється з opt-aiNet, IA-AIS та AAIS-2S у п'яти базових функціях та практичному застосуванні налаштування ПІД-регулятора. Згідно з результатами чисельного моделювання, як AIS-SSL, так і AIS-HSL можуть набагато швидше і точніше отримувати глобальні оптимальні або бажані рішення та ефективніші, ніж opt-aiNet, IA-AIS та AAIS-2S.

У статті [11] йдеться про меметичний алгоритм на основі клонального відбору, який використовується для вирішення проблем планування роботи цеху (Job-Shop Scheduling Problem). В запропонованому алгоритмі був використаний

принцип клонального відбору та розроблений метод локального пошуку для покращення розпізнавання. Для методу локального пошуку представлений імітаційний алгоритм, заснований на методі Новицького і Смутницького для отримання локального оптимуму. Запропонований алгоритм досліджується на деяких відомих задачах бенчмаркінгу. Чисельні результати підтверджують ефективність запропонованого алгоритму.

У статті [6] представлено застосування алгоритму клонального відбору в деяких задачах інженерії. Описана теоретична дія алгоритму, а також дія впровадженого алгоритму при вирішенні трьох різних проблем: бінарного розпізнавання символів, мультимодальної оптимізації функцій $-f(x,y) = x.\sin 4\pi x - y.\sin 4\pi y + 4 + 1$ і вирішення задачі комівояжера для 30 міст.

У статті [12] представлено використання алгоритму клонального відбору для оптимізації позиціонування будівельного майданчика. Представлений алгоритм мінімізує виробничі витрати і відстань, пройдену між n об'єктами, представленими за допомогою матриці перестановки розміру $n\ x\ n$.

У статті [13] представлена робота ШІС для вирішення ємнісної проблеми маршруту. Мета полягала в тому, щоб знайти правильні параметри алгоритму клонального відбору для вирішення проблеми за допомогою експериментального підходу. Крім ШІС, в статті також описується робота інших методів і представлено результати з точки зору якості рішення і часу розрахунків.

У статті [14] йдеться про використання алгоритму клонального відбору для визначення оптимальних робочих точок в низьковольтних гібридних мікромережах змінного або постійного струму. Мета полягала в мінімізації втрати активної потужності, експлуатаційних витрат та оптимізації напруги. Результати показують, що алгоритм прагне знайти найменше значення цільових функцій для різних початкових умов (наприклад, сезону, періоду дня, часу та завдання оптимізації). Коливання значень цільових функцій, що спостерігаються на початковій фазі алгоритму, є наслідком порушення обмежень. У завданнях оптимізації вузлового рівня напруги та мінімізації експлуатаційних витрат алгоритм стабілізується дуже

швидко, приблизно після 50 ітерацій. Завдання мінімізації втрат активної потужності є більш складним і вимагає близько 100 ітерацій.

Щоб обґрунтувати застосовність ШІС у процесі визначення оптимальних точок роботи гібридної низьковольтної мікромережі змінного/постійного струму, автор має намір протестувати алгоритм CLONALG з використанням інших цільових функцій (наприклад, мінімізації енергії, що імпортується з електромережі, максимізації кількості енергії, що виробляється у відновлюваних джерелах енергії та різних тестових мікромережах).

Таким чином, в оглянутих роботах науковці змогли досягти поставленої мети з використанням ШІС. Більшість авторів, підсумовуючи результати досліджень, наголошують, що через те, як працює алгоритм CLONALG, немає впевненості, що результати розрахунків є глобальним оптимумом цільових функцій. Тому в рамках подальших досліджень, щоб обґрунтувати застосовність ШІС в конкретних задачах оптимізації, варто порівняти отримані результати з результатами інших загальновживаних методів оптимізації.

1.5 Постановка цілі і задачі дослідження

Об’єктом дослідження є ШІС в задачах оптимізації. Предметом дослідження є використання штучної імунної системи для вирішення задачі оптимізації виробництва. Оптимізація полягатиме в пошуку параметрів моделі виробництва, які дають максимальний прибуток. Для моделювання роботи штучної імунної системи обраний алгоритм клонального відбору CLONALG.

В ході дослідження потрібно вирішити наступні задачі:

− проаналізувати предметну область;

− побудувати математичну модель виробництва і поставити задачу оптимізації;

− побудувати математичну модель штучної імунної системи, в основі якої лежатимуть принципи клонального відбору; адаптувати модель для вирішення задачі оптимізації змодельованого виробництва;

− розробити програмний додаток, який допоможе дослідити вплив параметрів побудованої ШІС на оптимізацію виробництва;

− проаналізувати отримані результати.

Необхідно провести тести, які покажуть вплив різних параметрів імунологічного алгоритму на результати оптимізації та обчислювальну складність. В випадку алгоритму CLONALG треба дослідити вплив розміру популяції, кількості ітерацій, коефіцієнту відібраних клітин, коефіцієнту клонування та сторожового таймеру. Необхідно довести збіжність алгоритму – властивість ітераційних алгоритмів досягати оптимуму цільової функції або достатньо близько підходити до нього за скінченне число ітерацій.

2 МАТЕМАТИЧНА МОДЕЛЬ ВИРОБНИЦТВА

2.1 Опис виробництва

Виробництво виготовляє деякі деталі. На початок дня на складі є сировина в певній кількості, а одиниця сировини придбана за певну ціну. Виробництво працює максимум 16 годин на день, одна повна зміна триває 8 годин. Якщо кількість робочих годин в певний день більше 8, то вважається, що підприємство працює в дві зміни. В певний день усі працівники можуть отримувати підвищену зарплату, отримуючи додатково до 50% заробітку. Також підприємство може працювати в прискореному темпі, виготовляючи деталі із швидкістю, вищою до 50% відсотків.

За день виробництво має виготовити певну кількість малих і великих деталей. При цьому в разі невиконання плану виробництво отримує штраф, який обчислюється по кількості невироблених малих і великих деталей.

На виробництві є певна кількість верстатів і пов’язані з ними оператори верстатів. Один оператор працює максимум 8 годин на день. Верстати є двох видів: верстати для виготовлення великих деталей і малих деталей. Для кожного типу верстату визначена заробітна плата оператора за годину роботи. Встановлена заробітна плата помножується на коефіцієнт бонусу, якщо такий є. Кожному типу машини потрібен свій час на підготовку до запуску. Для виготовлення однієї деталі потрібні встановлені кількість матеріалу і часу. Фактичний час виготовлення однієї деталі може бути меншим, якщо виробництво працює в прискореному темпі.

Виготовлена деталь, велика або дрібна, має базову вартість. Фактична вартість деталі може зрости, якщо оператор отримував бонус і/або виробництво працювало в прискореному темпі. В ході роботи верстат виготовляє певну кількість деталей за певний час. Деталі, вироблені зверх норми, є додатковим прибутком, проте якщо виготовлено забагато деталей, то вартість одиниці падає.

Прибуток виробництва – це сумарна вартість усіх виготовлених деталей за вирахуванням заробітної плати усім операторам і, при наявності, штрафу за невиготовлену кількість деталей.

На рисунку 5 схематично наведені сутності виробництва і відношення між ними.

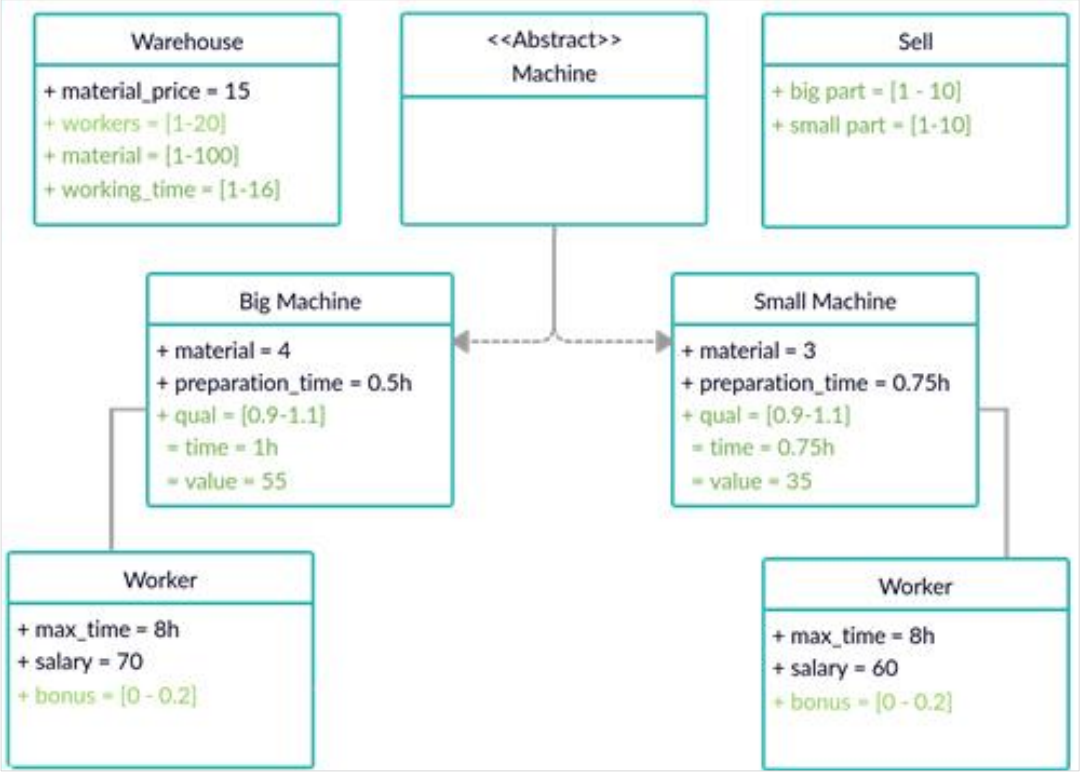


Рисунок 5 – Сутності виробництва

2.2 Функція цілі виробництва

Задача оптимізації полягає в пошуку максимального прибутку і формулюється наступним чином:

$$\text{Income} = \sum_{i=1}^{n_p} \left(p_i * \left(v_i - m_i * m_p\right)\right) - (1 + b_i) \times \times \sum_{i=1}^{n_w} (w_i * s_i * t_{wi}) - m_r * m_p) - \text{punish} \quad (1)$$

де n_p – кількість типів деталей (параметр), $p_i(n_m)$ – кількість деталей i-го типу, яка виробляється, $v_i(v_{bi}, t_{wi}, t_{bi}, w_q)$ – вартість деталей i-го типу, m_i – кількість

сировини, необхідна для виробництва деталі i-го типу (параметр), m_p – ціна сировини (параметр), b_i – премія працівнику (параметр), n_w – кількість розрядів (типів) працівників (параметр), w_i – кількість працівників i-го розряду (параметр), s_i – заробітна плата працівника i-го розряду, t_{wi} – фактичний час роботи машини або працівника i-го типу над одним продуктом, $m_r(p_i, n_m)$ – матеріал, що залишився, тобто невикористаний матеріал, $p_{i_{min}}$ – мінімальна кількість деталей, яку необхідно виготовити, щоб уникнути штрафу (параметр), $p_{i_{max}}$ – максимальна кількість деталей, яка може бути виготовлена, n_m – кількість сировини на складі на початок дня (параметр), $\text{punish}(p_{un}, p_{num}, v_i)$ – штраф за не вироблену необхідну кількість деталей

Штраф за невиконання необхідної кількості деталей обчислюється за формулою:

$$\text{punish} = p_{un} * \sum_{i=1}^{n_p}(p_{num_i}) * v_i, \tag{2}$$

$$p_{num_i} = \begin{cases} 0 & \text{if } p_{i_{min}} - p_i \leq 0 \\ p_{i_{min}} - p_i & \text{if } p_{i_{min}} - p_i > 0 \end{cases} \tag{3}$$

де p_{un} – штрафний коефіцієнт, $p_{num_i}(p_{i_{min}}, p_i)$ – кількість елементів i-го типу, для яких нарахований штраф, p_i – вироблена кількість деталей i-го типу

Кількість невикористаного матеріалу після закінчення виробництва обчислюється за формулою:

$$m_r = n_m - \sum_{i=1}^{n_p}(p_i * m_i) \tag{4}$$

Кількість працівників і-го типу дорівнює кількості машин і-го типу:

$$n_p = n_w \tag{5}$$

Необхідна кількість елементів і-го типу має задовольняти нерівності:

$$\sum_{i=1}^{n_p} p_{i_{max}} * m_i < n_m \tag{6}$$

Фактичний час роботи машини або працівника над одним виробом:

$$t_{wi} = t_{pi} + p_i * t_{bi} \tag{7}$$

Вартість однієї деталі обчислюється за формулою:

$$v_i = v_{bi} * (\frac{t_{wi}}{t_{bi}} * w_q * 100\%) \tag{8}$$

де v_{bi} – базова вартість продукту, виробленого машиною і-го типу (параметр), t_{bi} – базовий час роботи машини і-го типу над одним продуктом (параметр), w_q – якість працівника (залежить від бонусу)

Якість працівника обчислюється за формулою:

$$w_q = 1 + b * 0{,}4 \tag{9}$$

В таблиці 1 наведені параметри моделі виробництва, їх позначення і значення, використані в ході дослідження. В квадратних дужках подаються діапазони значень, в яких параметри приймають випадкове значення, а без дужок – константні значення.

Таблиця 1 – Параметри моделі

Параметр	Позначається як	Значення
Кількість матеріалу	n_m	$[x - 100x]$
Вартість матеріалу	m_p	4
Час праці	t_f	$[1 - 16]$
Мінімальна кількість великих деталей	$p_{0_{min}}$	$[0 - 10]$
Мінімальна кількість дрібних деталей	$p_{1_{min}}$	$[0 - 10]$
Зарплата оператора великого верстату	s_0	19
Кількість сировини на велику деталь	m_0	6
Час на підготовку великого верстату	t_{p_0}	1h 45 min
Вартість великої деталі	v_{b_0}	70
Час для виготовлення верстатом великої деталі	t_{b_0}	1h
Кількість великих верстатів	c_0	$[0 - 30]$
Зарплата оператора малого верстата	s_1	17
Кількість сировини на малу деталь	m_1	4
Час на підготовку малого верстату	t_{p1}	1h 25 min
Вартість малої деталі	v_{b1}	50
Час для виготовлення малої деталі	t_{b1}	1h 25 min
Кількість малих верстатів	c_1	$[0 - 30]$
Максимальний час праці працівника	t_w	8h
Премія працівника	b	$[0.0 - 0.5]$
Коефіцієнт штрафу	p_{un}	$[0 - 1]$

2.3 Алгоритм роботи виробництва

В даному підрозділі описано алгоритм роботи виробництва. Використано позначення, описані в таблиці 1 та розділі 4 – Програмна реалізація. Алгоритм складається з наступних етапів:

Крок 1 – ініціалізація фабрики.

Крок 2 – обчислити необхідну кількість матеріалу для вироблення замовлення, виражену виразом $p_0 * m_0 + p_1 * m_1$.

Крок 3 – якщо на складі недостатньо матеріалу – роботу алгоритму завершено, інакше робота алгоритму продовжується.

Крок 4 – обчислити чистий час роботи верстату і-го типу на день – час безпосереднього виробництва, в якому не врахований час на підготовку до роботи t_{p_0} або t_{p_1}.

Крок 5 – обчислити, скільки часу з урахуванням часу на підготовку необхідно верстатам і-го типу на виготовлення заданої кількості деталей:

Крок 5.1 – обчислити, за який час дана кількість верстатів зробить необхідну кількість деталей (з округленням в більшу сторону), врахувавши час на підготовку верстатів.

Крок 5.2 – якщо час > 8, значить робота займе дві зміни, і в часі на роботу двічі врахувати час на підготовку верстатів, так як з початком кожної зміни верстат перезапускається.

Крок 5.3 – якщо час < 8, то в часі на вироблення врахувати час на підготовку верстату одноразово.

Крок 6.1 – якщо повний час менший за час роботи виробництва на день t_f, тобто замовлення можливо виконати за один день, то витрачений час = результат кроку 5 і кількість вироблених деталей = required_parts

Крок 6.2 – інакше, якщо замовлення неможливо виконати за один день, то витрачений час встановлюється рівним

t_f і кількість вироблених деталей обчислюється як Округлення_Вниз(Чистий робочий час / Час на виготовлення однієї деталі) * Кількість_машин.

Крок 6.3 – віднімаємо витрачену кількість матеріалу з запасу матеріалу.

На цьому етапі завершується основний цикл роботи. Далі виправляються неточності в кількості вироблених деталей, що можливі через округлення, і починається цикл вироблення додаткових деталей.

Крок 7 – цикл з умовою: «ЯКЩО залишився матеріал І його достатньо для виготовлення хоча б однієї деталі великого АБО малого типу І у верстатів ще є робочий час, то створюємо додаткові деталі»

Перевіряємо, які верстати закінчили раніше – великі чи малі. Почати повинні ті, що закінчили раніше, щоб їм вистачило матеріалу, щоб виготовити деталі для виправлення невідповідності кількості вироблених деталей і витраченого часу.

Крок 7.1 – якщо є невідповідність між витраченим часом і кількістю виготовлених деталей під час першого циклу, то виправляємо її.

Крок 7.2 – інакше, якщо зробили необхідну кількість деталей за один пробіг і немає невідповідності, то йдемо на регулярний цикл вироблення додаткових деталей.

На цьому етапі закінчено вироблення деталей і далі обчислюється прибуток.

Крок 8.1 – для великих і малих верстатів перевіряємо: якщо кількість вироблених деталей менша за необхідну, то за кожну невироблену деталь нараховуємо штраф.

Крок 8.2 – підраховуємо сумарну вартість виготовлених деталей:

Крок 8.2.1 – якщо виробили занадто багато, то вартість однієї деталі падає.

Крок 8.2.2 – інакше залишається незмінною.

Крок 8.2.3 – сумарна вартість дорівнює вартості однієї деталі * кількість вироблених деталей.

Крок 8.3 – підраховуємо витрати на зарплату операторам великих і малих верстатів.

Крок 8.4 – прибуток дорівнює (Сумарна вартість великих деталей + Сумарна вартість дрібних деталей – Зарплата операторам великих машин – Зарплата операторам малих машин – Вартість матеріалу – Сумарний штраф.

Кінець роботи алгоритму.

Пояснення щодо кроку 6.1: на цьому етапі встановлюємо кількість виготовлених деталей, рівною required_parts, але витрачений час може бути встановлений більший, ніж необхідний для цієї кількості деталей. Це трапляється через округлення вгору у випадку, коли кількість деталей не ділиться порівну між верстатами, наприклад, *60 деталей / 8* верстатів = *7,5 деталей*, але через округлення вгору час підрахується для вироблення 8-ми деталей, і для 8-ми паралельно працюючих верстатів це повинно бути 64 деталі. Чому округлення і чому вгору? Округлити необхідно тому, що верстат може виробляти тільки цілу, закінчену деталь. Округлити вниз не можна, бо на цьому етапі перевіряються можливість вироблення заданої кількості деталей за робочий час. Якщо округлити вниз, то підрахується час для меншої кількості деталей, тут – для 56 деталей замість 60-ти.

Пояснення щодо кроку 6.2: округлюємо вниз, щоб отримати кількість цілих, завершених деталей, яку можуть виробити верстати за день. При цьому втрачається частина робочого часу.

Пояснення щодо кроку 7.2: на цьому етапі перевіряється, чи починається наступна зміна, чи вистачить часу на виготовлення однієї деталі і чи будуть задіяні усі верстати. Можлива ситуація, коли під кінець дня не вистачає часу, щоб виготовити одну деталь, в цьому випадку втрачаємо дарма частину часу. Обчислюємо, скільки деталей можна зробити з матеріалу, що залишився. Якщо кількість деталей, яку з нього можна виготовити, менша кількості верстатів, то не всі верстати будуть задіяні. Нерівність вигляду *spare_material / machine.mat_required < machine.machine_count.*

Алгоритм не враховує вартість палива або електроенергії для роботи верстатів, тому можна вважати, що всі верстати працюють до тих пір, поки хоча б один щось виробляє.

3 МОДЕЛЬ ШТУЧНОЇ ІМУННОЇ СИСТЕМИ

Для оптимізації фабрики обраний імунологічний алгоритм з використанням клонального відбору. Алгоритм був адаптований для вирішення поставленої задачі оптимізації виробництва. Клітиною, тобто антитілом, є пара «виробництво-прибуток». Афінністю антитіла вважатиметься прибуток даного виробництва. Чим вище прибуток виробництва, тим вища афінність антитіла. Використані позначення описані в розділах 2 і 4.

Алгоритм клонального відбору містить наступні параметри:

− кількість ітерацій;

− розмір популяції;

− коефіцієнт клонування;

− коефіцієнт відбору;

− сторожовий таймер – для зупинення процесу навчання, якщо немає прогресу.

Далі за допомогою псевдокоду описано алгоритм роботи ШІС для вирішення задачі оптимізації виробництва. Загальна послідовність кроків наступна:

```
clonalg():
1  population = generate_population(population_size)
  best = []
  for i in iterations:
2     selected = select(population, selection_rate)
3     clones = clone(selected, clone_rate)
4     matured = hypermutate(clones)
5     population = replace(population, matured)
6     best_value = max(population)
     best.append(best_value)
7       if i > watchdog:
           if best[-1] == best[-watchdog]:
              return best_value
```

Спочатку створюється випадкова популяція антитіл (крок 1). На кожній ітерації виконуються кроки 2-7, поки не закінчаться усі ітерації або роботу не зупинить сторожовий таймер.

Створення клонів (крок 3):

```
clone(selected, clone_rate):
    clone_number = clone_rate * population_size
    # offset = найбільше абсолютне значення
    offset = max( abs(min(selected), max(selected)) + 1
    val_sum = sum( (sel.value + offset) for sel in selected)
    clones = []
    while clone_number > 0:
        for cell in selected:
            factor = (cell.value + offset) / val_sum
            clones += [cell for i in range(ceil(factor *
clone_number)]
            clone_number -= math.ceil(factor * clone_number)
    return clones
```

Кількість клонів пропорційна афінності клітини. Знаходимо найбільше абсолютне значення – мінімальне або максимальне значення афінності. Це число буде зсувом, щоб уникнути ділення на нуль. Обчислюємо коефіцієнт, який показує, скільки клонів від загальної кількості створюваних клонів буде створено для даної клітини.

Мутація клонів (крок 4) відбувається наступним чином:

```
hypermutate(clones):
  max_value = max(clones)
  for clone.factory in clones:
    mutation_factor = get_mutation_factor(max_value,clone.value)
    for each prop of factory
      mature_parameter(prop.value, mutation_factor)
return clones
```

Сила мутації обернено-пропорційна афінності клону: чим гірше афінність, тим інтенсивніша мутація. Знаходимо значення найкращої афінності серед клонів. Для кожного клону обчислюємо коефіцієнт мутації, мутуємо кожне поле об’єкта виробництва.

Коефіцієнт мутації обчислюється наступним чином:

```
get_mutation_factor(max_value,                                value):
    return abs((max_value - value) / max_value * 0.6 + 0.1)
```

Далі показано алгоритм зміни параметрів виробництва:

```
mature_parameter(current_value, mutation_factor):
    # отримати діапазон зміни значення
    val_range = ceil((max_value - min_value) * mutation_factor)
    # отримати випадкове значення з діапазону
```

```
    mutation_value = random(-val_range, val_range)
    # перевірити, чи отримане значення належить діапазону
    if current_value + mutation_value > max_value:
        return max_value
    elif current_value + mutation_value < min_value:
        return min_value
    return current_value + mutation_value
```

Якщо отримане значення більше максимально допустимого, встановлюємо найбільше допустиме значення. І навпаки, якщо менше мінімального значення, встановлюємо найменше допустиме значення.

Далі на кроці 6 відбувається заміна клітин вихідної ітерації кращими мутованими клонами, як показано в псевдокоді:

```
replace(population, matured):
    new_population = population, max_value = max(matured)
    considered_p = []
    for cell in population:
        if cell.value < max_value: considered_p.add(cell)
    if considered_p.length < round(sel_rate*population.length):
        how_many_changes = considered_p.length
    else:
        how_many_changes = round(selection_rate * population.length)
    already_changed = []
    while already_changed.length < how_many_changes:
        choosen = -1 if already_changed.length == 0 else
already_changed[0]
        while choosen in already_changed or choosen == -1:
            # get index of cell that will be changed
            choosen = choice(considered_p)
already_changed.append(choosen)
        considered_m = []
        for mature in matured:
            if mature.value > sorted(considered_p)[choosen].value:
                considered_m.append(mature)
        index_to_change = new_population.index(
            sorted(considered_p)[choosen])
        new_population[index_to_change] = considered_m[randint(0,
considered_m.length - 1)]
    return new_population
```

Ймовірність заміни залежить від значення. Знаходимо значення найкращої афінності. Додаємо до масиву considered_p клітини, значення яких менше

максимального. Обчислюємо необхідну кількість замін. Створюємо масив already_changed. Поки він менше how_many_changes, заміняємо гіршу частину популяції мутованими клонами. Отримаємо індекс клітини для заміни:

```
choice(population):
    cdf_vals = cdf(population.length, b=1.5)
    idx = bisect(cdf_vals, random())
    return idx
cdf(how_many, b):
    # коефіцієнти отримані з кв.рівняння з заданим параметром b:
a*x^2 + b*x + c = 0
    assert 2 > b > 1
    result = []
    for i in range(1, how_many):
        result.add((1 - b) * (i / how_many) ** 2 + b * (i /
how_many))
    return result
```

Пояснення щодо способу отримання індексу клітини для заміни. Мета в тому, щоб обрати одну клітину із популяції, але необхідно, щоб імовірність вибору була різною для різних клітин. Необхідно отримати розподіл, який давав би більше шансів обрати менші індекси (вибірка упорядкована за зростанням). Щоб це зробити, не можна взяти просто випадковий індекс (це був би «дискретний рівномірний розподіл»). Розглянемо частину квадратного рівняння, зображену на рисунку 6.

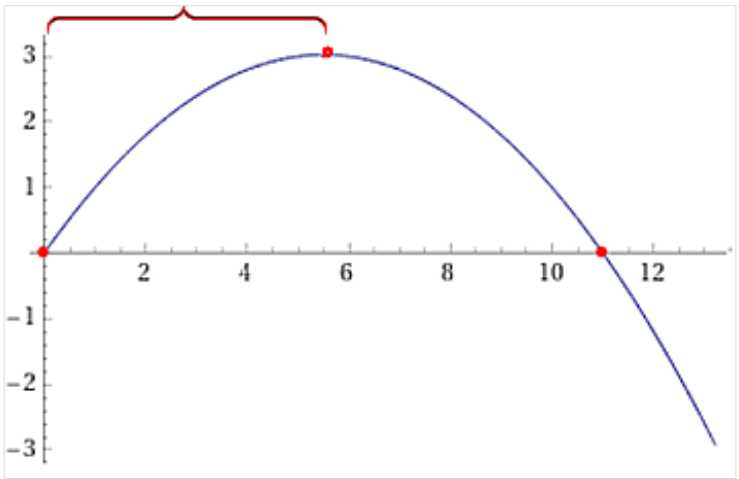


Рисунок 6 – Графік квадратного рівняння

Припустимо, що популяція складається з трьох клітин. І ми маємо квадратичну функцію: $y = -0.5x^2 + 1.5x$ (далі буде пояснено, чому функція

такого вигляду). Розрахуємо 4 значення: $y(0{,}2) = 0{,}28$, $y(0{,}4) = 0{,}52$, $y(0{,}6) = 0{,}72$, $y(0{,}8) = 0{,}88$.

Ці значення ділять діапазон $(0, 1)$ на 5 частин різного розміру, де кожна наступна частина менша за попередню. Тепер за допомогою цього поділу можна випадково задати значення від 0 до 1, а за допомогою функції bisect.bissect() можна отримати відповідний індекс. Наприклад, візьмемо випадкове значення 0,51, тоді bisect поверне 1 (перший індекс, ми обрали клітину з першим індексом, щоб мутувати її). Тому в цьому прикладі ми просто змінили рівномірний розподіл на нерівномірний, так що шанси обрати наступні індекси змінилися з $[0.2, 0.2, 0.2, 0.2, 0.2]$ на $[0.28, 0.24, 0.2, 0.16, 0.12]$.

Чому квадратична функція і чому b знаходиться в діапазоні $(0{,}1)$. Візьмемо квадратичну функцію на проміжку від 0 до 1, яка задовольняє $y(0) = 1$ і $y(1) = 1$. З першого отримуємо $c = 0$, а з другого $a = 1 - b$.

Таким чином, отримуємо вираз $(1 - b) * x^2 + b * x$. Маючи таку функцію, необхідно визначити, яке b вона може прийняти. Якщо $b < 1$, тоді парабола перевертається, і шанси вибрати менший індекс стають меншими, ніж вибір більшого, тому це непідходяще b. Якщо $b > 2$, то вершина x-компонента параболи зміщується між 0 і 1, що може призвести до ситуації, коли наш поділ виглядає, як $[0.3, 0.5, 0.65, 0.7, 0.65, 0.5, 0.3]$, тому значення спочатку йдуть вгору, а потім вниз, і це також не той результат, якого необхідно досягти.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Загальний опис

Програмний додаток створено на мові програмування Python. Дана мова підтримує кілька парадигм програмування, зокрема об'єктно-орієнтовану та функціональну. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також зручним засобом поєднання наявних компонентів.

Додаток створено в середовищі PyCharm. Перевагами цього інструменту є наявність спеціальних інструментів для візуалізації даних, інтелектуальні засоби рефакторингу коду та вбудована консоль, а також безкоштовна ліцензія для здобувачів вищої освіти.

Для розробки програмного додатку підключені наступні модулі:

− Matplotlib.pyplot – модуль для побудови графіків, який працює схожим чином з MATLAB;

− PPrint – модуль для структурованого виводу об’єктів на консоль;

− Logging – модуль для ведення журналу транзакцій додатку;

− Abc – модуль для створення абстрактних класів;

− Math – модуль із математичними функціями;

− Random – модуль для отримання псевдовипадкових значень;

− Bisect – модуль, що забезпечує підтримку списка в упорядкованому стані за допомогою алгоритму ділення навпіл.

Програмний додаток складається із модулів, які за своїм призначенням об’єднані в пакети:

− Logic – пакет, який містить бізнес-логіку та математичні обчислення;

− Models – пакет, який містить моделі сутностей предметної області;

− Utils – пакет із допоміжними модулями.

Файл main.py – точка входу у програму. Config.py – файл, в якому користувач додатку задає параметри моделі та алгоритму.

На рисунку 7 наведений приклад налаштування моделі виробництва та штучної імунної системи.

config.py

```
# ALGORITHM SETTINGS
iterations = 200
population_size = 100
clone_rate = 0.5
selection_rate = 0.5
# Watchdog for breaking learning process if no  progress
watchdog = 50


# FACTORY SETTINGS
max_big_machines = 30
max_small_machines = 30
min_working_time = 1
max_working_time = 16
max_bonus = 0.5
max_haste = 0.5
requirements = {'req_big': 50, 'req_small': 40,
                'big_punish': 40, 'small_punish': 35}


# CONSTANTS
material_cost = 4
big_spec = {'prep_time': 1.75, 'runtime': 1.5,
            'product_value': 70, 'mat_required': 6, 'base_salary': 19}
small_spec = {'prep_time': 1.25, 'runtime': 1.25,
              'product_value': 50, 'mat_required': 4, 'base_salary': 17}

# RANDOMS
min_material = big_spec['mat_required'] * requirements['req_big'] + \
               small_spec['mat_required'] * requirements['req_small']
max_material = min_material * 100
```

Рисунок 7 – Програмний код файлу config.py

В секції ALGORITHM SETTINGS задаються налаштування алгоритму клонального відбору:

– кількість ітерацій;

– розмір популяції;

– коефіцієнт клонування;

– коефіцієнт відбору;

– сторожовий таймер – для зупинення процесу навчання, якщо немає прогресу.

В секції FACTORY SETTINGS задаються граничні значення для випадково згенерованих вхідних параметрів моделі фабрики:

− максимальна кількість великих верстатів на виробництві;

− максимальна кількість малих верстатів на виробництві;

− мінімальний час роботи виробництва на добу;

− максимальний час роботи виробництва на добу;

− максимальний бонус працівникам;

− максимальне прискорення робочого процесу;

− вимоги щодо кількості виготовлених деталей та штраф за невиконання.

В секції CONSTANTS задаються вихідні дані виробництва:

− вартість одиниці матеріалу;

− час на запуск верстата;

− час, необхідний на виготовлення однієї деталі;

− вартість виготовленої деталі;

− кількість матеріалу, необхідна для виготовлення однієї деталі;

− заробітна плата оператора верстата.

В секції RANDOMS по введеним вище значенням вираховуються мінімальна і максимальна кількість матеріалу, необхідна виробництву на день роботи.

Пакет Utils містить модулі, які виконують допоміжну роботу. Модуль logger.py записує результати експериментів в журнал транзакцій. Зібрані дані можуть бути використані для подальшого аналізу. Factory_logs.log – журнал подій додатку.

Модуль plot_drawer.py призначений для побудови графіків, інкапсулює роботу із pprint та matprolib.pyplot. Побудовані графіки можна переглянути прямо в середовищі PyCharm. За допомогою вбудованого інструменту SciView можна маніпулювати графіком або експортувати його як зображення. Якщо програма запущена із провідника, то після закінчення роботи консольного додатку відкриється вікно із побудованим графіком та інструментами для роботи з ним.

Модуль factory_model_test.py використовується для тестування фабрики. За введеними параметрами будується об’єкт фабрики і виводиться на екран. Можна

побачити, як модель коригує невалідні параметри (наприклад, якщо передане значення коефіцієнту більше за максимальне, то встановлюється найвище можливе) і показує значення обчислювальних полей.

4.2 UML-моделювання

На рисунку 8 зображена UML–діаграма моделі виробництва. Модель складають класи, які описують сутності предметної області.

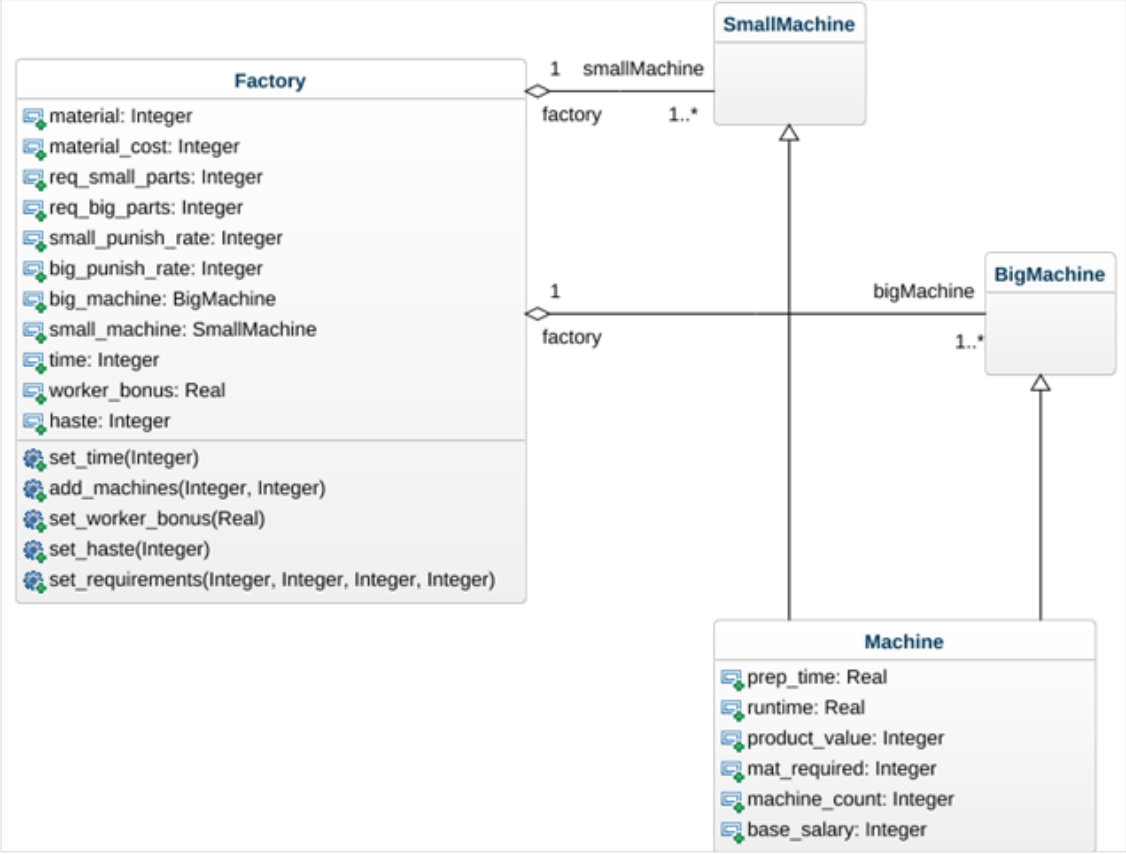


Рисунок 8 – UML-діаграма моделі виробництва

До класів моделі відносяться:

− Factory – клас, який представляє виробництво; згідно діаграмі, на одному виробництві може використатися кілька верстатів;

− Machine – абстрактний клас, який описує загальні риси верстата;

− BigMachine – клас, який наслідується від Machine і представляє верстат для вироблення великих деталей;

− SmallMachine - клас, який наслідується від Machine і представляє верстат для вироблення дрібних деталей.

Властивості класу Factory:

− material – кількість матеріалу на виробництві на початок дня;

− material_cost – ціна одиниці матеріалу;

− req_small_parts – кількість дрібних деталей, які необхідно виробити за день;

− req_big_parts – кількість великих деталей, які необхідно виробити за день;

− small_punish_rate – штраф за невироблення необхідної кількості дрібних деталей;

− big_punish_rate – штраф за невироблення необхідної кількості великих деталей;

− big_machine – великі верстати виробництва;

− small_machine – малі верстати виробництва;

− time – кількість робочих годин виробництва на день;

− worker_bonus – бонус працівникам за вироблену кількість деталей вищу за заплановану;

− haste – коефіцієнт прискорення виробництва.

Властивості верстатів:

− prep_time – час на запуск верстату;

− runtime – час, за який верстат виготовляє одну деталь;

− product_value – вартість однієї деталі, виготовленої верстатом;

− mat_required – кількість матеріалу, необхідна для виготовлення однієї деталі;

− machine_count – кількість одночасно працюючих верстатів певного типу;

− base_salary – базова зарплата оператора верстату певного типу.

4.3 Приклади коду і програмний вивод

Пакет Logic містить бізнес-логіку і обчислення предметної області. Модуль factory_service реалізує логіку моделі, тобто імітує роботу виробництва. В модулі CLONALG реалізовані обчислення моделі штучної імунної системи.

Наведемо програмну реалізацію функції вибору антитіла з популяції для заміни мутованим клоном:

```
def cdf(how_many, b):
    # weights calculated on the basis of the parabola of the
quadratic equation with the given parameter b: a*x^2 + b*x + c =
0
    # b should be within the range (1,2)
    assert 2 > b > 1
    result = []
    for i in range(1, how_many):
    result.append((1 - b) * (i / how_many) ** 2 + b * (i /
how_many))
    return result
def choice(population):
    cdf_vals = cdf(len(population), 1.5)
    x = rd.random()
    idx = bisect.bisect(cdf_vals, x)
    return idx
```

В даній реалізації обраховується функція дискретного розподілу, щоб отримати якнайменший індекс списку, детально обчислення описані в розділі 2.

Приклад програмної реалізації функції коригування невідповідності кількості виготовлених деталей і витраченого часу в випадку, коли неможливо рівномірно поділити роботу між машинами:

```
def first_cycle(machine, spare_material):
        if spare_material >= machine.mat_required * (
                machine.machine_count - machine.parts_required %
machine.machine_count):
            used_material = machine.mat_required * (
                        machine.machine_count -
machine.parts_required % machine.machine_count)
            machine.created_parts += (machine.machine_count -
machine.parts_required % machine.machine_count) return
used_material
            used_material = machine.mat_required *
(spare_material // machine.mat_required)
            machine.created_parts += spare_material //
machine.mat_required return used_material
```

Код додаткового циклу вироблення деталей, коли зроблена основна робота і на виробництві ще є час і матеріал:

```
def regular_cycle(machine, spare_material, max_time):
        if spare_material // machine.mat_required <
machine.machine_count:
            if machine.elapsed_time < 8 and machine.elapsed_time
+ machine.real_runtime > 8:
                if machine.elapsed_time + machine.real_runtime +
machine.prep_time < max_time:
machine.elapsed_time += machine.real_runtime + machine.prep_time
machine.created_parts += spare_material // machine.mat_required
used_material = (spare_material // machine.mat_required) *
machine.mat_required
                else:
                    machine.elapsed_time = max_time
                    used_material = 0
            else
  if machine.elapsed_time + machine.real_runtime < max_time:
  machine.elapsed_time += machine.real_runtime
  machine.created_parts+= spare_material // machine.mat_required
  used_material = (spare_material // machine.mat_required) *
 machine.mat_required
                 else:
                     machine.elapsed_time = max_time
                     used_material = 0
             return used_material
```

Код пропорційного клонування кращих антитіл:

```
def select(population, x):
    # select <x> best cells
    selected = sorted(population, key=lambda population:
population['value'], reverse=True)[:x]
    return selected
def clone(selected, clone_rate):
    # clone cells, clones count proportional to cell's value
    clone_number = clone_rate * population_size
    # offset = biggest absolute value
    offset = max(abs(selected[0]['value']), abs(selected[-
1]['value']))+1
    val_sum = sum([sel['value'] + offset for sel in selected])
    clones = []
    while clone_number > 0:
        for cell in selected:
            factor = (cell['value'] + offset)/val_sum
            clones += [cell for i in range(math.ceil(factor *
clone_number))]
            clone_number -= math.ceil(factor * clone_number)
    return clones
```

На рисунку 9 наведений приклад побудованого графіка цільової функції від кількості ітерацій (робота створеного модулю plot_drawer).

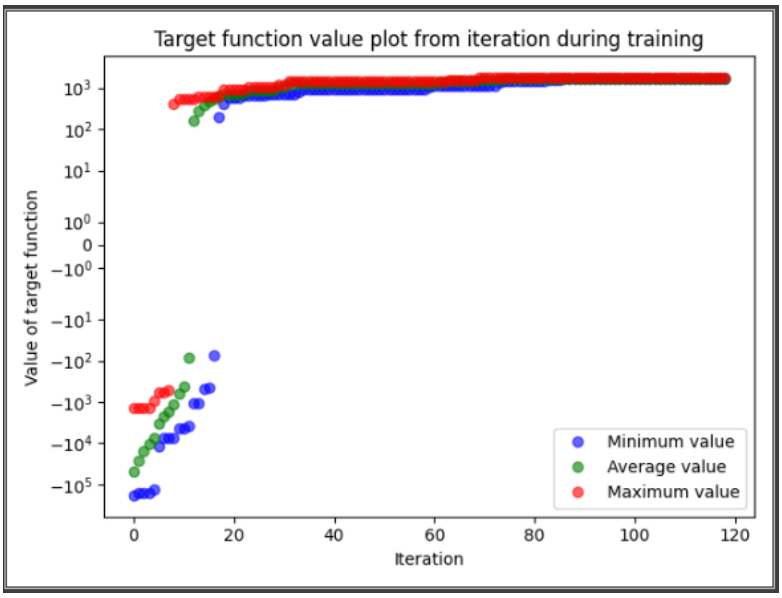


Рисунок 9 – Приклад побудованого графіку цільової функції

Модуль дозволяє не тільки будувати за даними графики, а й виводити їх у розгорнутому вигляді на консоль, як показано на рисунку 10.

```
Iteration: 174  Best value: 1756.87 Worst value: 1703.69    Average value: 1730.28
Iteration: 175  Best value: 1756.87 Worst value: 1703.69    Average value: 1730.28
Iteration: 176  Best value: 1756.87 Worst value: 1703.69    Average value: 1730.28
Iteration: 177  Best value: 1756.87 Worst value: 1703.69    Average value: 1730.28
Iteration: 178  Best value: 1756.87 Worst value: 1703.69    Average value: 1730.28
Iteration: 179  Best value: 1756.87 Worst value: 1703.69    Average value: 1730.28
Max value 1756.87. No progress since 50 iterations
{'object': <Models.factory.Factory object at 0x0000022B58AEE130>,
 'specifications': {'big_machines': 12,
                    'bonus': 0.2799999999999999,
                    'haste': 0,
                    'material': 943,
                    'small_machines': 9,
                    'time': 16},
 'value': 1756.87}

Process finished with exit code 0
```

Рисунок 10 – Приклад програмного виводу на консоль

Таким чином, в даному розділі розглянуто загальний зміст програмного додатку, описана робота допоміжних модулів, побудована UML-діаграма моделей предметної області і наведено приклади коду бізнес-логіки і програмного виводу.

5 ДОСЛІДЖЕННЯ ВПЛИВУ ПАРАМЕТРІВ МОДЕЛІ ШТУЧНОЇ ІМУННОЇ СИСТЕМИ НА РЕЗУЛЬТАТИ ОПТИМІЗАЦІЇ

5.1 Опис експерименту

Проведено серію тестів для різних параметрів алгоритму. Параметрами за замовчуванням були: розмір популяції = 100, кількість ітерацій = 200, коефіцієнт відібраних клітин = 20%, коефіцієнт клонування = 50% та watchdog = 50. Окремі параметри змінюються для визначення їх впливу на результат оптимізації. Найкращий розв’язок – це максимальне значення прибутку, отримане за 100 запусків алгоритму. Параметри моделі виробництва наведені в розділі 2.

5.2 Отримані результати

5.2.1 Вплив параметрів на процес оптимізації

Досліджено вплив розміру популяції. В таблиці 2 наведені результати роботи алгоритму оптимізації виробництва з різними значеннями розміру популяції.

Таблиця 2 – Вплив розміру популяції

Розмір популяції	Графік	Найкраще значення	Ітерація з найкращим значенням	Найкращий розв’язок
10	Графік 2	1748.8	74	1931.39
40	Графік 3	1875.88	20	
200	Графік 4	1917.62	55	
500	Графік 5	1931.39	94	

Найкращий розв’язок – 1931.39, отриманий на 94 ітерації для максимального розміру популяції – 500 антитіл.

В таблиці 3 наведені результати роботи імунологічного алгоритму при різних значеннях кількості ітерацій (мається на увазі кількість ітерацій в алгоритмі, а не кількість запусків алгоритму).

Таблиця 3 – Вплив кількості ітерацій

Кількість ітерацій	Графік	Найкраще значення	Ітерація з найкращим значенням	Найкращий розв’язок
10	Графік 6	1632.2	10	1907.22
40	Графік 7	1871.88	37	
100	Графік 8	1904.13	73	
500	Графік 9	1907.22	91	

Найкращий розв’язок – 1907,22 отриманий для конфігурації з 500 ітераціями на 500-й ітерації.

В таблиці 4 наведені результати роботи алгоритму в залежності від значення коефіцієнту клонування.

Таблиця 4 – Вплив коефіцієнту клонування

Коефіцієнт клонування	Графік	Найкраще значення	Ітерація з найкращим значенням	Найкращий розв’язок
0,25	Графік 10	1819,73	26	1915,86
0,40	Графік 11	1847,17	113	
0,60	Графік 12	1915,86	172	
0,80	Графік 13	1904,72	74	

Найкращий розв’язок – 1915,86, отриманий на 172-й ітерації з коефіцієнтом клонування 0,60.

В таблиці 5 наведені результати роботи алгоритму в залежності від значення контролюючого таймера (кількість ітерацій, після яких алгоритм зупинить роботу за відсутності прогресу).

Таблиця 5 – Вплив контролюючого таймера (watchdog)

Таймер	Графік	Найкраще значення	Ітерація з найкращим значенням	Найкращий розв’язок
25	Графік 14	1870.86	19	1915.15
40	Графік 15	1915.15	158	
60	Графік 16	1908.33	199	
80	Графік 17	1897,93	151	

Найкращий розв’язок – 1915.15, отриманий на 158-й ітерації з контролюючим таймером 80.

В таблиці 6 наведені результати роботи алгоритму в залежності від значення коефіцієнту вибраних клітин.

Таблиця 6 – Вплив коефіцієнту вибраних клітин

Коефіцієнт вибраних клітин	Графік	Найкраще значення	Ітерація з найкращим значенням	Найкращий розв’язок
0,05	Графік 18	1899,58	74	1906,54
0,10	Графік 19	1906,54	24	
0,40	Графік 20	1905,00	57	
0,50	Графік 21	1899,50	46	

Найкращий розв’язок – 1906,54, отриманий на 24-й ітерації з коефіцієнтом клонування 0,40.

Отримані дані придатні для подальшого аналізу. Далі наведемо графіки значень цільової функції для кожної ітерації алгоритму.

5.2.2 Графіки значень цільової функції

Побудовані графіки показують значення цільової функції (максимізація прибутку) від номеру ітерації алгоритму. Для параметрів за замовчуванням найкращий результат – 1917,62 за 183 ітерації. На рисунку 1 наведений графік для результатів із параметрами за замовчуванням.

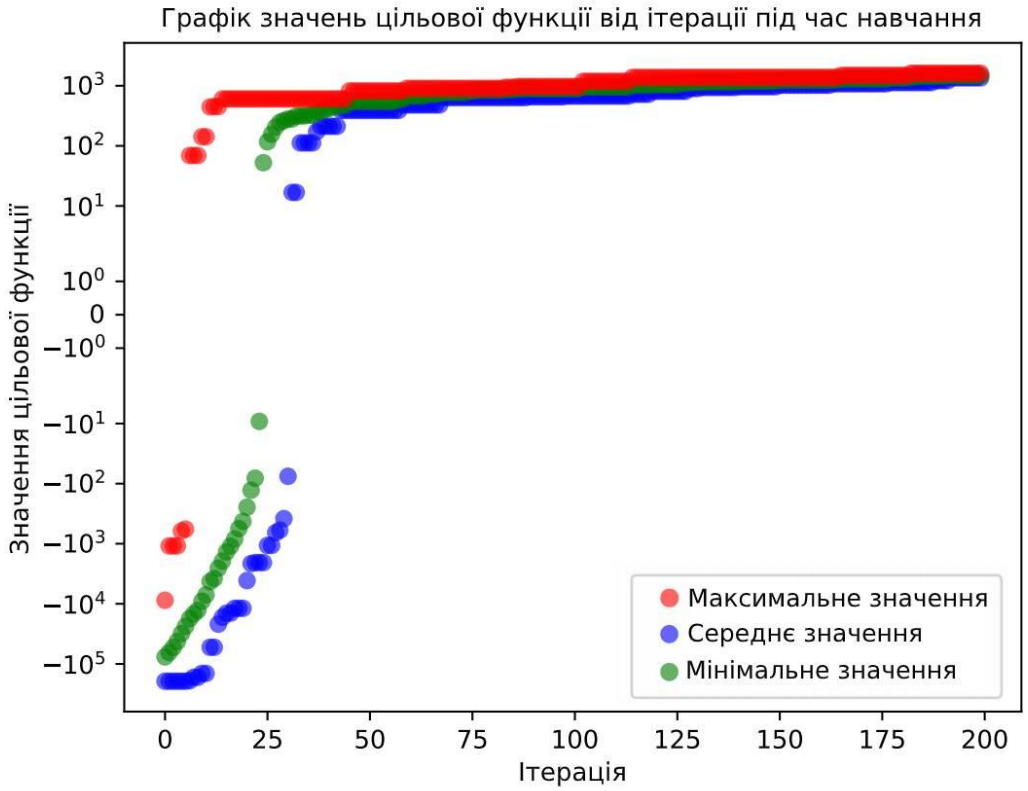
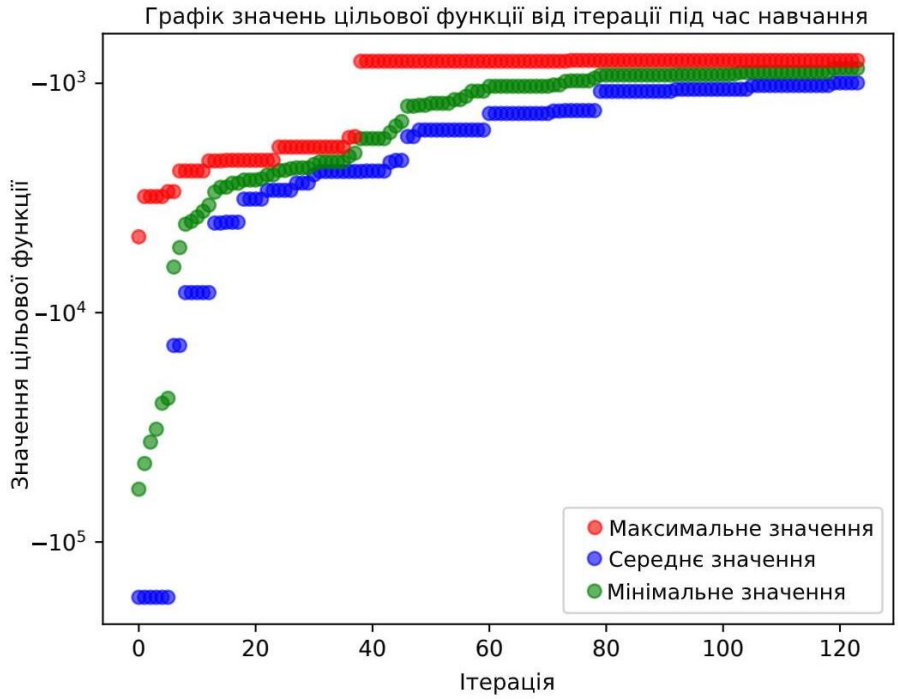
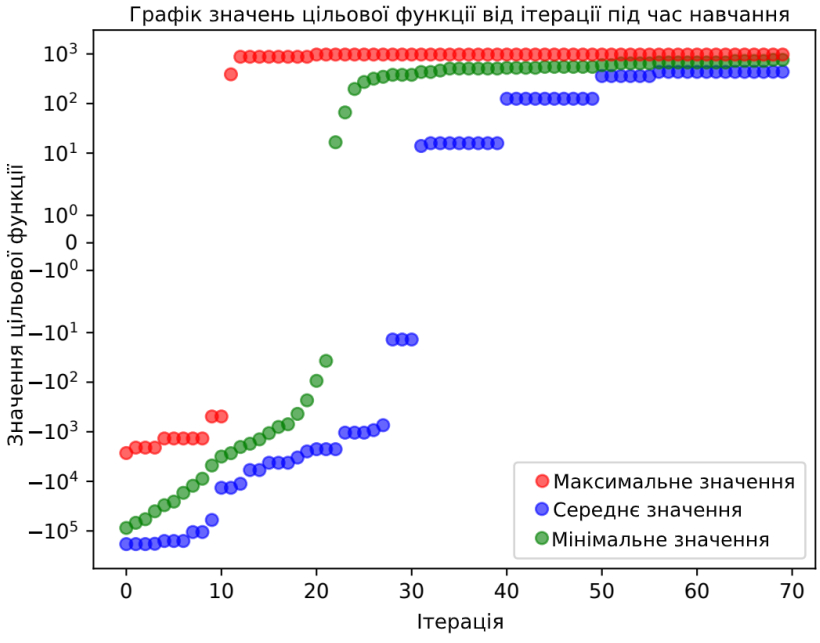


Рисунок 11 – Запуск з параметрами за замовчуванням

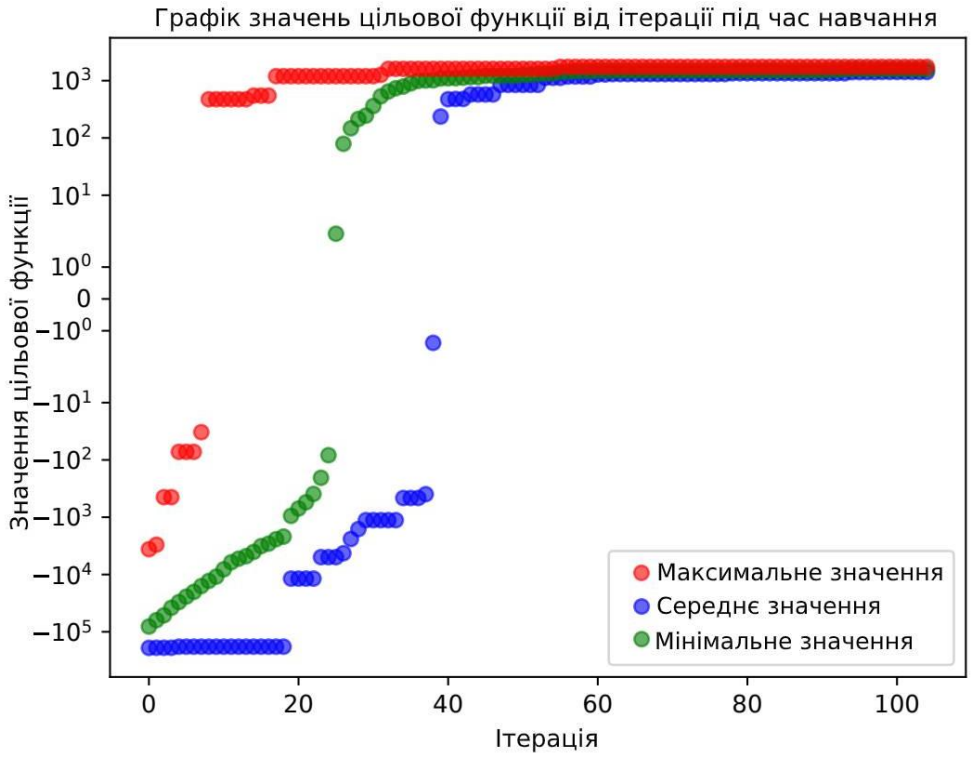
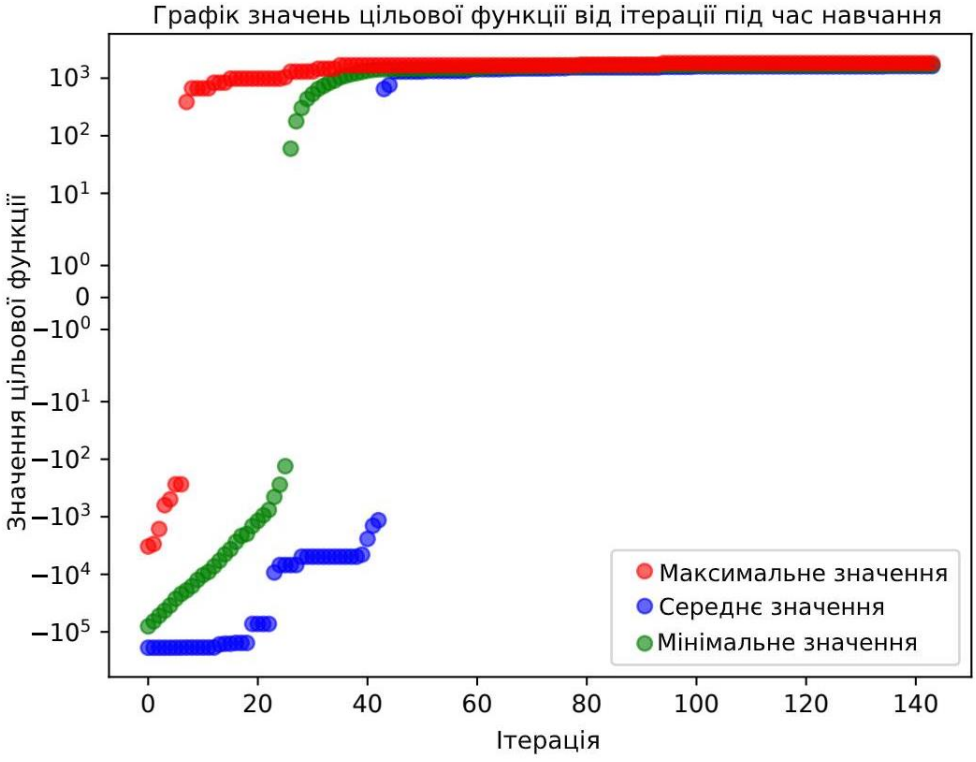
На рисунку 11 і далі по вісі абсцис знаходяться номери ітерацій, а по вісі ординат – значення цільової функції. Червоним кольором зображуються максимальні значення на ітерації, синім – середні і зеленим – мінімальні значення. Так як програма запускає алгоритм 100 разів для визначення максимального найкращого розв’язку, побудовано графіки для запуску, на якому отримано з найкраще значення.

Далі наведено графіки значень цільової функції для запуску алгоритму зі зміненням значень окремих параметрів моделі штучної імунної системи.

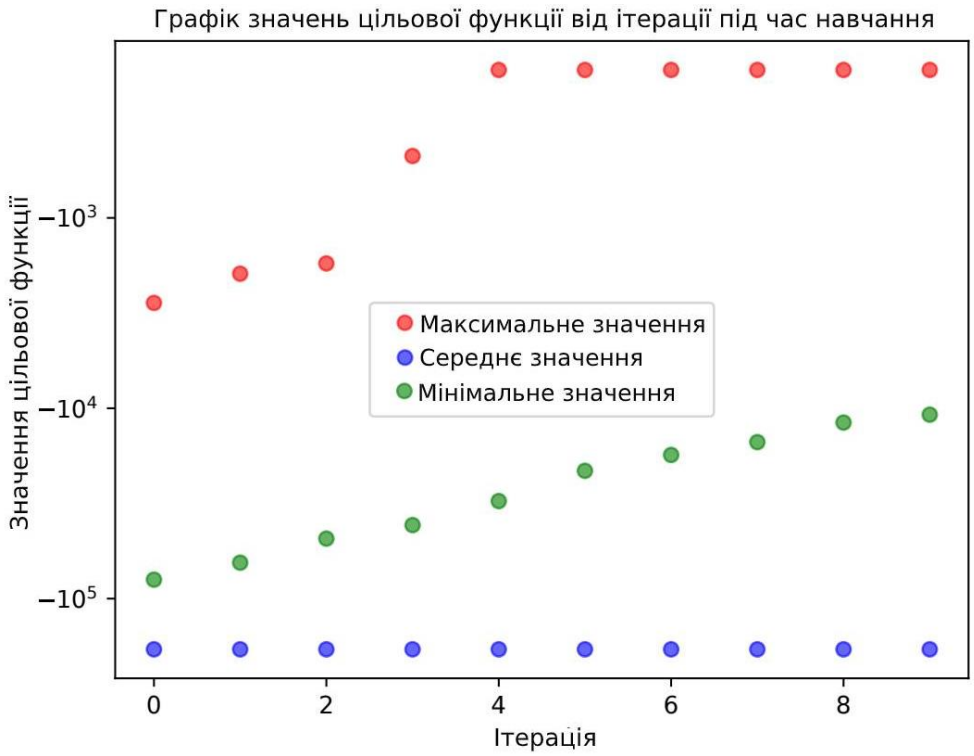
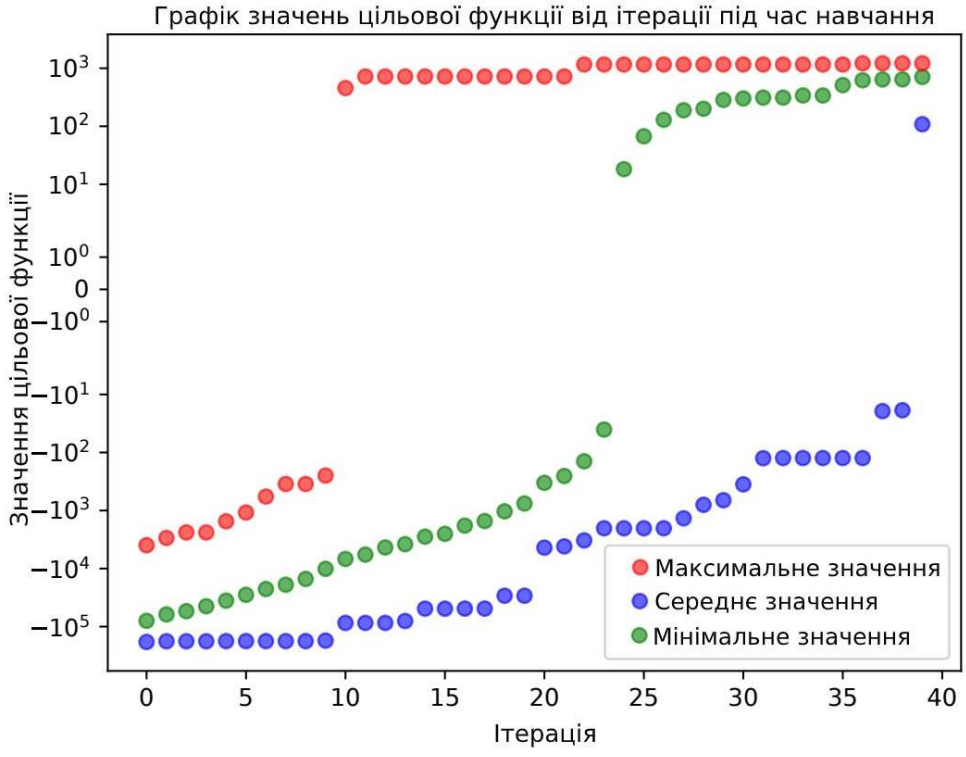
Таблиця 7 – Вплив розміру популяції

№	Графік значень цільової функції від ітерації під час навчання	Розмір популяції
2		10 клітин
3		40 клітин

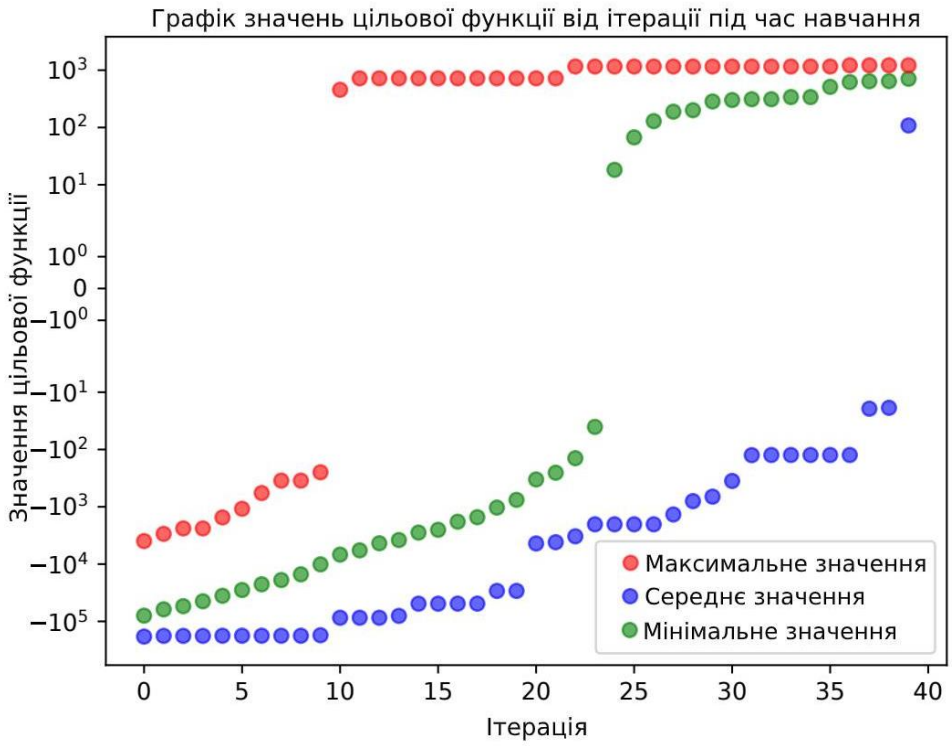
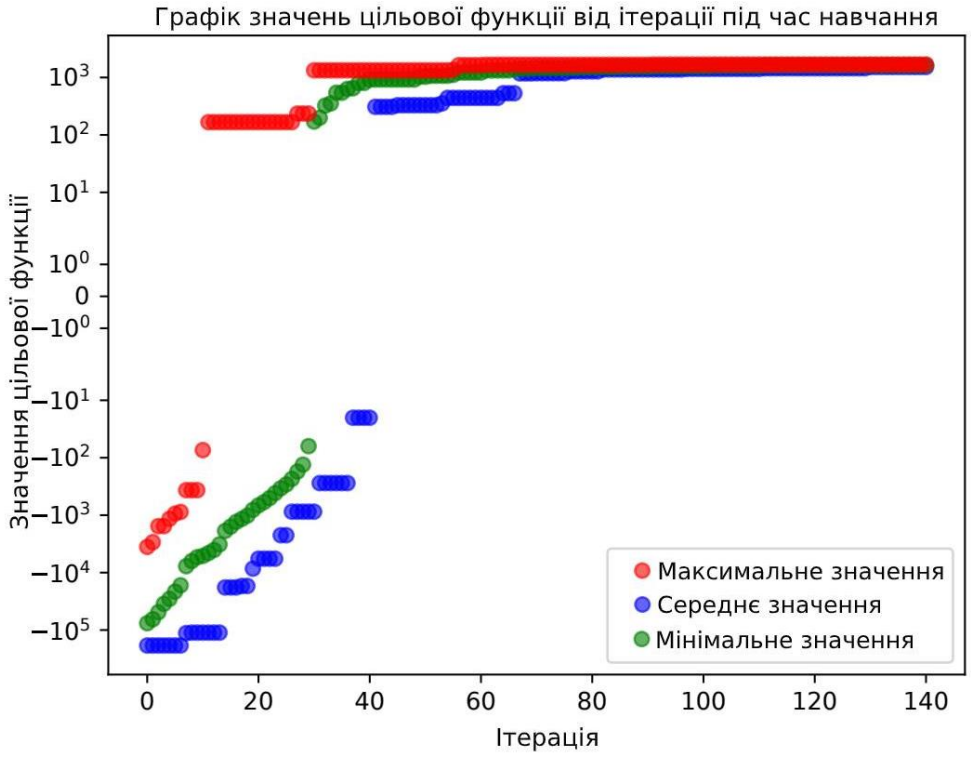
Продовження таблиці 7

№	Графік значень цільової функції від ітерації під час навчання	Розмір популяції
4		200 клітин
5		500 клітин

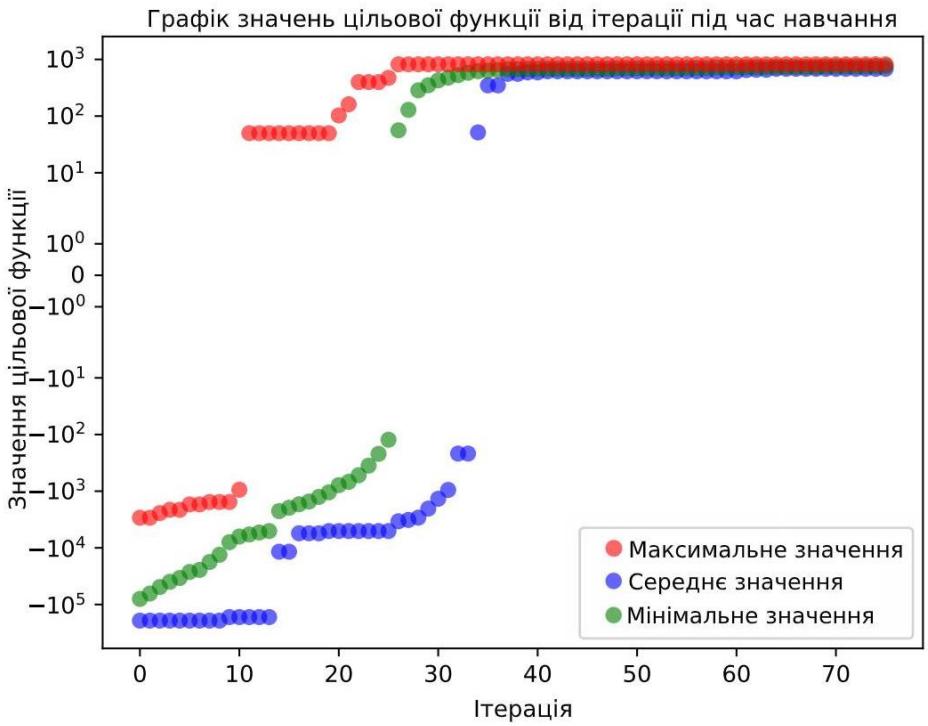
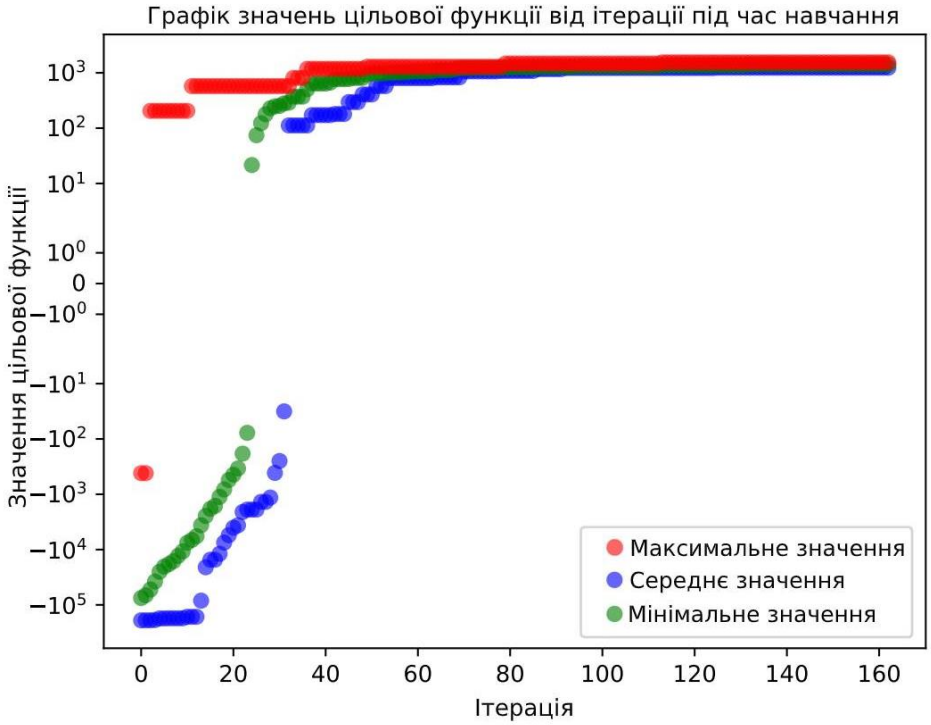
Таблиця 8 – Вплив кількості ітерацій

№	Графік значень цільової функції від ітерації під час навчання	Кількість ітерацій
6		10
7		40

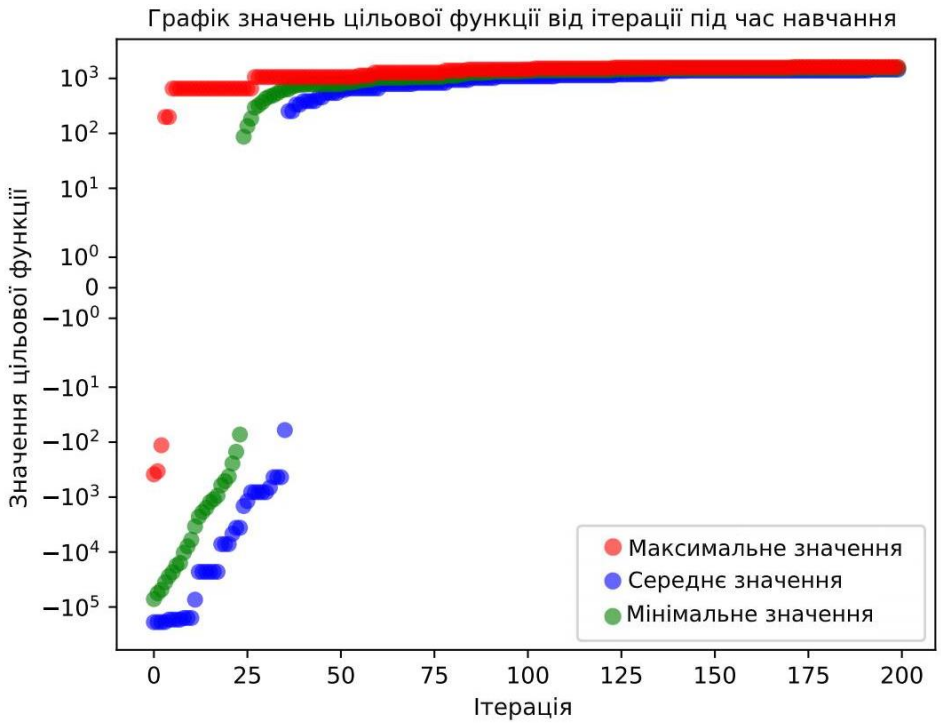
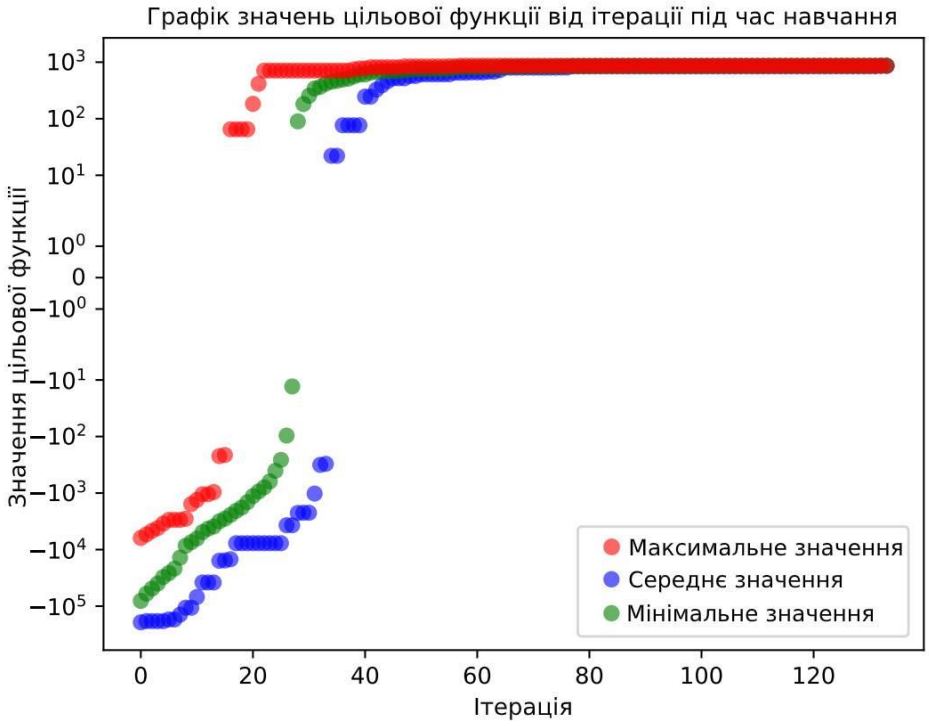
Продовження таблиці 8

№	Графік значень цільової функції від ітерації під час навчання	Кількість ітерацій
8		100
9		500

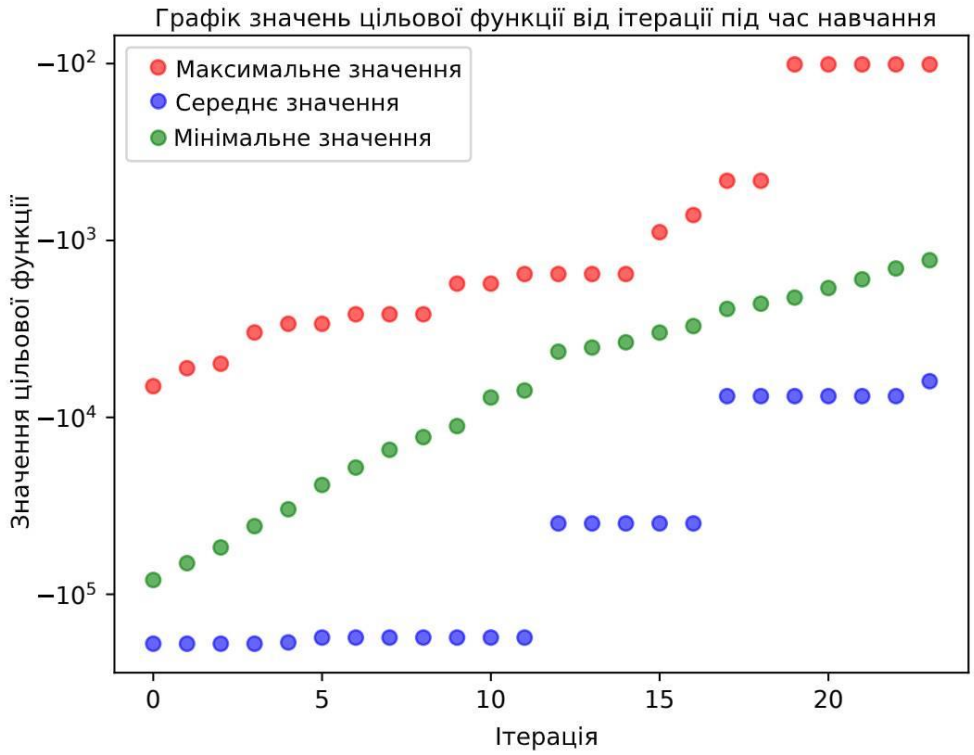
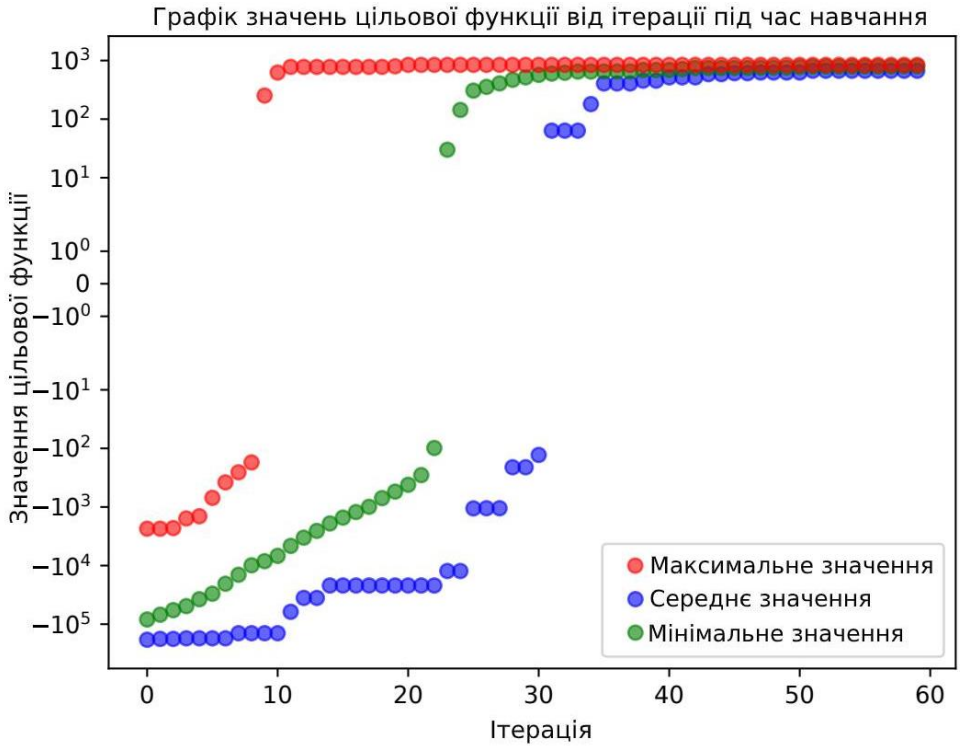
Таблиця 9 – Вплив коефіцієнту клонування

№	Графік значень цільової функції від ітерації під час навчання	Коефіцієнт клонування
10		25%
11		40%

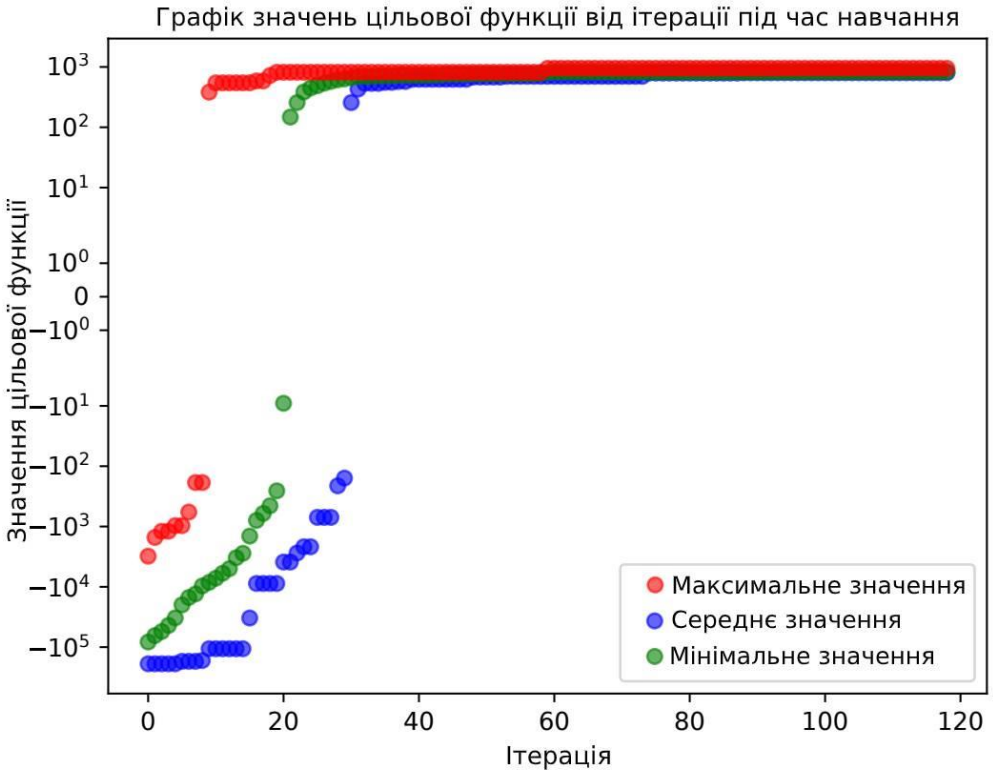
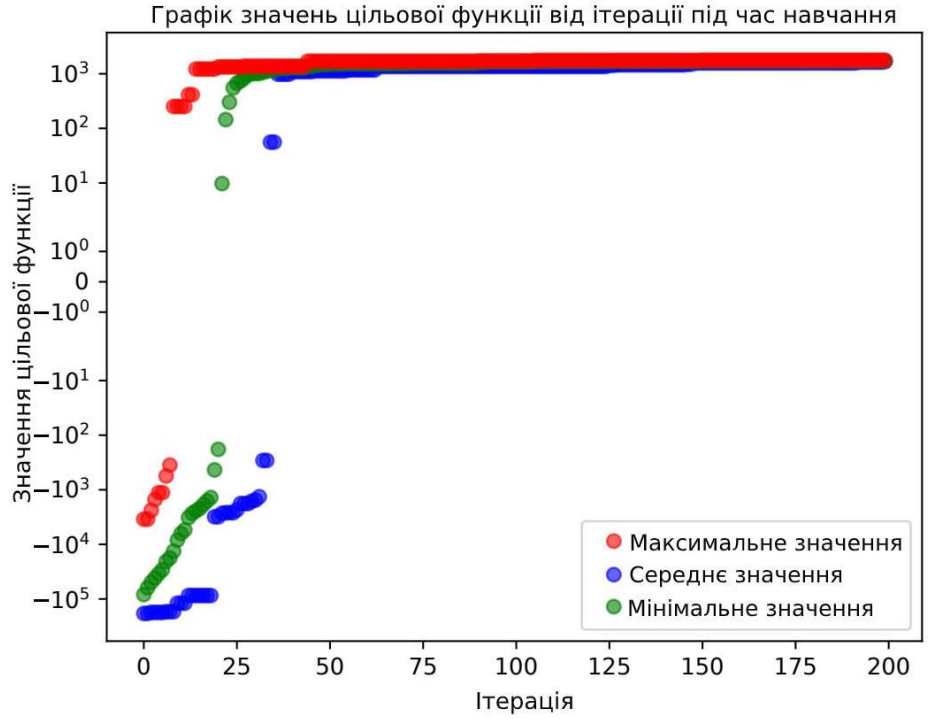
Продовження таблиці 9

№	Графік значень цільової функції від ітерації під час навчання	Коефіцієнт клонування
12		60%
13		80%

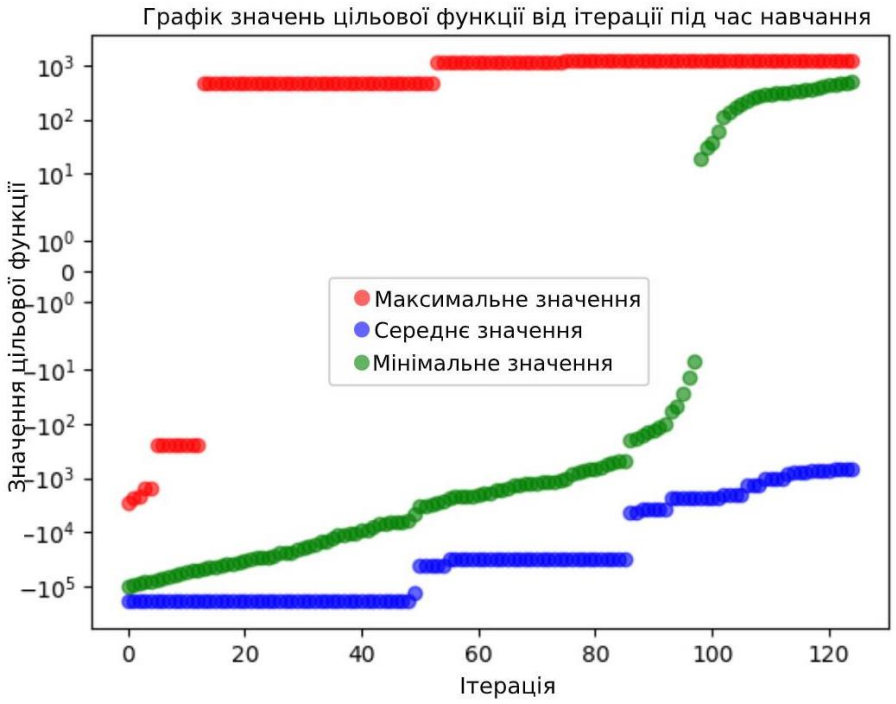
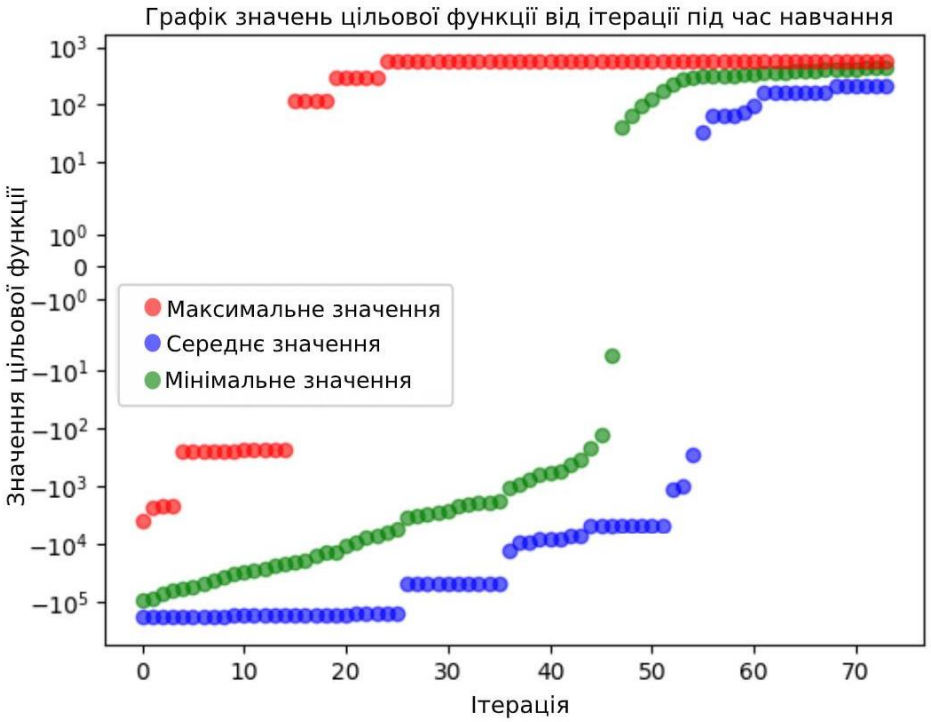
Таблиця 10 – Вплив сторожового таймера

№	Графік значень цільової функції від ітерації під час навчання	Watchdog
14		5
15		40

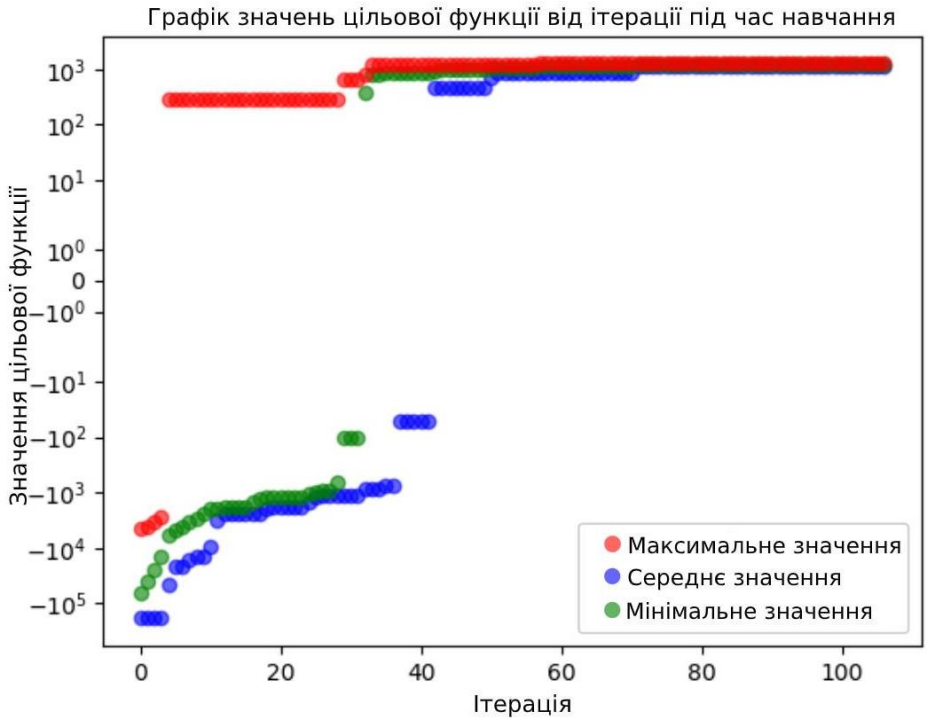
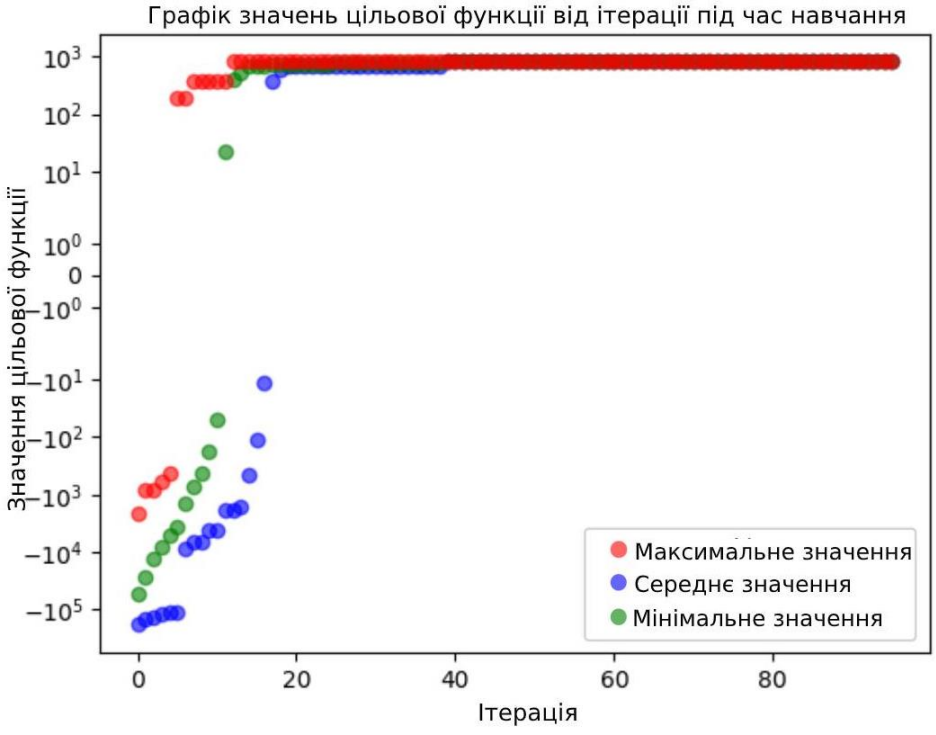
Продовження таблиці 10

№	Графік значень цільової функції від ітерації під час навчання	Watchdog
16		60
17		80

Таблиця 11 – Вплив коефіцієнту обраних клітин

№	Графік значень цільової функції від ітерації під час навчання	Коефіцієнт обраних клітин
18		5%
19		10%

Продовження таблиці 11

№	Графік значень цільової функції від ітерації під час навчання	Коефіцієнт обраних клітин
20		40%
21		50%

5.3 Аналіз результатів експерименту

5.3.1 Вплив розміру популяції

На основі проведених тестів можна зробити висновок, що розмір популяції сильно впливає на результати моделювання. Зі збільшенням розміру популяції значно покращується найкращий можливий результат. Чим більша популяція, тим більше рішень задачі порівнюються одночасно, що, при відсутності змін інших параметрів, природно збільшує шанси знайти краще рішення. Крім того, видно, що для невеликих груп алгоритм закінчувався до 200-ї (значення за замовчуванням) ітерації. Це може бути тому, що вся популяція "застряє" на локальному мінімумі і не може з нього вийти (тому що на підставі інших тестів відомо, що кращі рішення існують). Щоб уникнути подібних ситуацій у майбутньому, коефіцієнт мутації повинен бути краще підібраний.

5.3.2 Вплив кількості ітерацій

На підставі тестів можна зробити висновок, що кількість ітерацій суттєво не впливає на рішення. В тестах, проведених після певної кількості ітерацій, було видно, що найкраще значення залишається незмінним. Отже достатньо, щоб кількість ітерацій була досить великою, щоб алгоритм дійшов до точки, коли значне поліпшення зупиняється.

Таким чином, ймовірно, було б краще, якби кількість ітерацій залежала від інших параметрів. Те, що кількість ітерацій безпосередньо не впливає найкраще значення, отримане за допомогою алгоритму, можна підтвердити порівнянням тесту на 40 і 100 ітерацій. При 40 ітераціях рішення «впали» в кращий локальний мінімум, і найкращий результат був близько 1200, тоді як при 100 ітераціях рішення

"впали" в гірший локальний мінімум, і після досягнення межі 800, незважаючи на велику кількість ітерацій, вони не змогли суттєво покращитися.

5.3.3 Вплив коефіцієнту клонування

З тестів випливає, що коефіцієнт клонування певною мірою впливає на якість рішення. Природно, що чим більше клонів, тим більше мутацій, тим більша ймовірність появи розв'язків кращих, ніж попередніх. Однак і обчислювальна складність суттєво зростає. Отже, параметр слід вибирати оптимально з точки зору підвищення якості рішення та часу виконання алгоритму. Окрім того, із збільшенням коефіцієнту покращення якості, схоже, знижується, що дозволяє припустити, що відповідний оптимум існує.

Найкраще значення для коефіцієнту клонування 0.8, яке менше найкращого значення для коефіцієнту 0.6, здається випадковістю, а не зниженням якості рішення внаслідок зростання коефіцієнту. До таких висновків можна дійти, спираючись на аналіз роботи всього алгоритму, де збільшення коефіцієнта не може послабити якість найкращого рішення.

5.3.4 Вплив зміни сторожового таймера

На основі тестів можна визначити, що вплив параметра сторожового таймера малий. Даний параметр впливає на результат алгоритму лише при малих його значеннях, коли алгоритму не вдається виправити рішення за кілька кроків, а сторожовий таймер досить малий, щоб припинити алгоритм до досягнення максимуму значення. Під час багатьох тестів не вдалося побачити ситуацію, в якій найкраще рішення не змінювалось більше 40 ітерацій, а потім раптом значно

покращилось. Таким чином, можна зробити висновок, що для даного алгоритму встановлювати значення параметра watchdog більше 40 не має сенсу і лише подовжує час роботи алгоритму.

5.3.5 Вплив змін коефіцієнта обраних клітин

Видно, що зміна параметра не має суттєвого впливу на якість найкращого рішення. Проте це видно і по самому алгоритму. Оскільки алгоритм замінює вихідні клітини лише найкращими рішеннями, вибір додаткових, гірших рішень для створення клонів, які, мабуть, і так буде опущено, був би невиправданим.

ВИСНОВКИ

В ході виконання атестаційної роботи була досліджена предметна область штучних імунних систем, зокрема в задачах оптимізації. Було створено модель виробництва і сформульовано задачу оптимізації – підвищення прибутку. Для максимізації цільової функції був використаний імунологічний алгоритм CLONALG, який було адаптовано для вирішення поставленої задачі. Моделі виробництва і штучної імунної системи були реалізовані в програмному додатку, за допомогою якого було проведене дослідження.

В результаті вдалось досягти поставленої мети дослідження. Тести показали, що залежно від різних налаштувань параметрів алгоритм здатний знайти найкращий результат за кінцеву кількість ітерацій. Хоча через свій спосіб роботи алгоритм не завжди досягає глобального оптимуму в кожному конкретному запуску, отримані значення цільової функції задовільні.

Досліджено вплив таких параметрів ШІС: розмір популяції, кількість ітерацій, коефіцієнт клонування, сторожовий таймер, коефіцієнт обраних клітин. В підрозділі 5.3 детально описаний вплив параметрів на результати оптимізації.

Для малої кількості ітерацій та розміру популяції алгоритм є відносно ефективним з точки зору вартості обчислень, але іноді це обертається гіршою якістю отриманих результатів. З іншого боку, для вищих значень цих параметрів можна чітко дослідити, що обчислювальна вартість більша, однак це призводить до отримання кращих результатів.

В подальших дослідженнях буде доцільно протестувати алгоритм CLONALG з іншими цільовими функціями (наприклад, мінімізації витраченого часу на роботу, мінімізації витрат на матеріал). Також варто порівняти отримані результати з результатами інших загальновживаних методів оптимізації.

Підводячи підсумок, імунологічний алгоритм, що використовує принцип клонального відбору, виявився здатним вирішити поставлену задачу оптимізації виробництва та може бути успішно використаний і для інших задач оптимізації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Nicholson, L.B. (2016). The Immune System. Essays Biochem, (60): 275-301. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5091071/

2. Chaplin, D. (2010). Overview of the Immune Response. J Allergy Clin Immunol., (125), 3-23. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2923430/

3. Dasgupta, D. & Forrest, S. Artificial Immune Systems in Industrial Applications: Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials. (Vol. 1, pp 257–267). July 10-15, 1999, Honolulu, USA.

4. Korablev N.M. & Ivaschenko G.S. (2014). Parallel Immune Algorithm of Short-Term Forecasting Based on Model of Clonal Selection. Radio Electronics, Computer Science, Control, (2), 73-78.

5. Jansen, T. & Zarges C. (2014). Reevaluating Immune-Inspired Hypermutations Using the Fixed Budget Perspective. IEEE Transactions on Evolutionary Computation, (18.5), 674–688.

6. Castro, L. N. & Zuben, F. (2002). The Clonal Selection Algorithm with Engineering Applications 1st. GECCO 2002 – Workshop Proceedings. Retrieved from: https://www.researchgate.net/publication/2468677_The_Clonal_Selection_Algorithm_with_Engineering_Applications

7. Castro, L. N. & Zuben, F. (2002). Learning and Optimization Using the Clonal Selection Principle. IEEE Transactions on Evolutionary Computation, (6), 239-251. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2923430/

8. Корабльов М.М., Макогон А.Е. & Фомічов О.О. (2011). Аналіз збіжності імунних алгоритмів. Системи обробки інформації, 29-33. 14 лютого, 2011, Харків, Україна.

9. Campelo, F. (2005). A Clonal Selection Algorithm For Optimization In Electromagnetics. IEEE Transactions on Magnetics, (41), 1736-1739.

10. Wang, M. (2017). An Artificial Immune System Algorithm with Social Learning and Its Application in Industrial PID Controller Design. Mathematical Problems in Engineering, (2017), 1–13. Retrieved from https://www.hindawi.com/journals/mpe/2017/3959474/

11. Yang, J. (2008). Clonal Selection Based Memetic Algorithm For Job Shop Scheduling Problems. Journal of Bionic Engineering, (5.2), 111–119. Retrieved from https://www.sciencedirect.com/science/article/abs/pii/S1672652908600141

12. Wang, X. (2016). Application of Clonal Selection Algorithm in Construction Site Utilization Planning Optimization. Procedia Engineering. Integrating Data Science, Construction and Sustainability, (146), 267–273. Retrieved from http://www.sciencedirect.com/science/article/pii/S1877705816300789

13. Thapatsuwan, W.,Pongcharoen, P. & Thapatsuwan, P. (2013). Clonal Selection of Artificial Immune System for Solving the Capacitated Vehicle Routing Problem. Journal of Next Generation Information Technology, (4), 167–179. Retrieved from https://www.researchgate.net/publication/269787610_Clonal_Selection_of_Artificial_Immune_System_for_Solving_the_Capacitated_Vehicle_Routing_Problem

14. Rokicki, L. (2019). The Application Of The CLONALG Algorithm In The Process Of Optimal Operation Control Of Hybrid AC/DC Low Voltage Microgrid. E3S Web of Conferences 8. Retrieved from https://www.e3s-conferences.org/articles/e3sconf/pdf/2019/10/e3sconf_pe2019_02011.pdf