**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

"___"_____2016 р.

# Дипломна робота

першого (бакалаврського) **рівня вищої освіти**
(першого (бакалаврського), другого (магістерського))

зі спеціальності 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код та назва спеціальності)

на тему: Алгоритми та програмні засоби обробки зображень на основі генетичних алгоритмів оптимізації

Виконав (-ла): студент (-ка) 4 курсу, групи ДА-21
(шифр групи)

Лешик Андрій Вікторович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник к.т.н., доц. каф. СП ІПСА Чкалов О.В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Економічний д.е.н., проф. Семенченко Н.В. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль ст. викладач Британов О.А.

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2016 року

**Національний технічний університет України**
**«Київський політехнічний інститут»**

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський)
(перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2016 р.

**ЗАВДАННЯ**
**на дипломний проект (роботу) студенту**
Лешику Андрію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Алгоритми та програмні засоби обробки зображень на основі генетичних алгоритмів оптимізації,

керівник проекту (роботи) к.т.н., доц. каф. СП ІПСА Чкалов О.В.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 12 травня 2016 р. № 50-ст

2. Строк подання студентом проекту (роботи) 09.06.2016

3. Вихідні дані до проекту (роботи):

Зображення формату Bitmap (*.bmp)
Розміри зображення: до 1,5 мегапікселя
Форма реалізації – у вигляді програми на мові С++. Можливе використання додаткових бібліотек для роботи із зображеннями.

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

1. Розглянути можливі варіанти реалізації ядра еволюційних обчислень в контексті обробки зображень.
2. Розробити ядро еволюційних обчислень.
3. Розробити план тестування ядра еволюційних обчислень.
4. Протестувати ядро еволюційних обчислень.
5. Розробити алгоритм обробки зображень, використовуючи ГА.
6. Виконати програмну реалізацію та провести тестування.
7. Проаналізувати роботу програмного продукту на наборі тестових зображень.
8. Виконати економічний аналіз програмного продукту.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)

1. Блок-схема алгоритму роботи ГА – плакат.
2. Алгоритм обробки зображень, використовуючи ГА – плакат.
3. Порівняльна таблиця моделей ГА – плакат.
4. Результати тестування програмного продукту – плакат.

6. Консультанти розділів проекту (роботи)*

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

7. Дата видачі завдання 01.02.2016

Календарний план

| № з/п | Назва етапів виконання дипломного проекту (роботи) | Строк виконання етапів проекту (роботи) | Примітка |
|---|---|---|---|
| 1 | Отримання завдання | 01.02.2016 | |
| 2 | Збір інформації | 15.02.2016 | |
| 3 | Вивчення варіантів реалізації та вибір варіанту для розробки | 28.02.2016 | |
| 4 | Розробка ядра ГА | 10.03.2016 | |
| 5 | Розробка плану тестування | 12.03.2016 | |
| 6 | Тестування ядра ГА | 14.03.2016 | |
| 7 | Розробка алгоритму обробки зображень | 15.03.2016 | |
| 8 | Виконання програмної реалізації | 01.04.2016 | |
| 9 | Аналіз роботи програмного продукту | 01.05.2016 | |
| 10 | Аналіз економічної обґрунтованості | 15.05.2016 | |
| 11 | Оформлення дипломної роботи | 31.05.2016 | |
| 12 | Отримання допуску до захисту та подача роботи в ДЕК | 09.06.2016 | |

* Консультантом не може бути зазначено керівника дипломного проекту (роботи).

# АНОТАЦІЯ


бакалаврської дипломної роботи Лешика Андрія Вікторовича

на тему «Алгоритми та програмні засоби обробки зображень

на основі генетичних алгоритмів оптимізації»

Дана дипломна робота присвячена розробці програмного продукту для обробки зображень. Метою роботи є розробка програмного модуля, що базується на генетичних алгоритмах.

В роботі розглянуто загальну модель генетичних алгоритмів, розроблено алгоритм для покращення візуальної якості зображень, проведено тестування алгоритму використовуючи різні моделі еволюційних обчислень, проведено аналіз результатів та на їх основі створено програмний продукт. Наведено мінімальні та оптимальні параметри для запуску програмного забезпечення.

Загальний обсяг роботи: 81 сторінка, 15 рисунків, 10 таблиць, 36 бібліографічних найменувань.

Ключові слова: еволюційні обчислення, генетичні алгоритми, обробка зображень, оптимізація, крайові детектори.

# АННОТАЦИЯ

бакалаврской дипломной работы Лешика Андрея Викторовича

на тему «Алгоритмы и программные средства обработки изображений

на основе генетических алгоритмов оптимизации»

Данная дипломная работа посвящена разработке программного продукта для обработки изображений. Целью работи есть разработка программного модуля, базирующегося на генетических алгоритмах.

В работе рассмотрена общая модель генетических алгоритмов, разработан алгоритм для улучшения визуального качества изображений, проведено тестирование алгоритма используя различные модели эволюционных вычислений, проведен анализ результатов и на их основе создан программный продукт. Приведены минимальные и оптимальные параметры для запуска программного обеспечения.

Общий объем работы: 81 страница, 15 рисунков, 10 таблиц, 36 библиографических наименований.

Ключевые слова: эволюционные вычисления, генетические алгоритмы, обработка изображений, оптимизация, краевые детекторы.

# ANNOTATION

a bachelor’s degree work of Andrii Leshyk

entitled "Algorithms and Software Tools for Image Processing Based on Genetic Optimization Algorithms"

This project is devoted to the research of software tools for image processing. The aim is to develop an image-processing tool based on genetic algorithms.

The project covers the following: basic genetic algorithms model; developed algorithm for image processing based on genetic algorithms; efficiency tests for different genetic algorithms models; minimal and optimal launch parameters for developed program.

Total volume of work: 81 pages, 15 figures, 10 tables, 36 bibliographic titles.

Keywords: evolutionary computation, genetic algorithms, image processing, optimization, edge detectors.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

| | |
|---|---|
| ПЗ | Програмне забезпечення |
| ГА | Генетичні алгоритми |
| ОК | Оператор кросинговеру |
| ОМ | Оператор мутації |
| КП | Крайові пікселі |
| ЯЕО | Ядро еволюційних обчислень |
| ЕО | Еволюційні обчислення |

# ВСТУП

Використання цифрових зображень почалося ще на початку 1920-х років[1]. З появою комп’ютерів у середині XX століття виникла нова область науки – цифрова обробка зображень. З’явилися перші задачі з обробки цифрових зображень: стиснення зображень, покращення якості, сегментація, морфологічний аналіз та багато інших.

Характерною особливістю багатьох методів цифрової обробки зображень є проблема формалізації перетворень. Спроби вирішувати дані задачі детерміністичними методами часто накладало різні обмеження, наприклад, складність формального представлення перетворень, наявність великої кількості змінних та відсутність еталонного розв’язку. Саме тому в процесі розвитку методів обробки зображень почали з’являтися методи, основою яких є нові концепції пошуку рішень в обчислювальних системах. Однією з таких концепцій є еволюційні обчислення.

Важливою перевагою генетичних алгоритмів є поєднання стохастичних та детермінованих властивостей. Використання генетичних алгоритмів для пошуку екстремумів складних багатомодальних функцій дає можливість пошуку рішень за прийнятний час та з достатньо високою точністю [2, 3, 4, 5]. Крім того, використання ГА в таких задачах дозволяє знаходити рішення навіть при невідомій формі перетворення, наприклад при підборі структури й ваг нейронної мережі при нейро-еволюційному підході покращення зображень [6]. Для роботи ГА достатньо можливості формального представлення рішень та критерію оцінки результату. При цьому, за рахунок стохастичної складової ГА мають доволі широкий простір пошуку, а за рахунок механізму наслідування дозволяють зберегти властивості близьких до оптимальних рішень.

Таким чином, можна зробити висновок про актуальність дослідження методів обробки зображень з використанням еволюційних обчислень.

Метою роботи є розробка повністю автоматичного алгоритму покращення цифрових зображень, побудованого на методах еволюційних обчислень, а також його реалізація у вигляді програмних засобів.

Для досягнення поставленої мети необхідно послідовне вирішення таких задач:

1. Загальний аналіз методів еволюційного підходу до вирішення задач оптимізації та формування на основі результатів аналізу вимог до генетичного алгоритму для обробки зображень.
2. Вибір та дослідження операторів та моделі генетичного алгоритму, які найбільше підходять для вирішення задач обробки цифрових зображень.
3. Розробка алгоритму обробки зображень на основі ГА з урахуванням результатів попередніх кроків. Дана задача також включає у себе аналіз ефективності використання розробленого алгоритму.
4. Розробка та адаптація додаткових етапів обробки для підвищення швидкості та якості обробки. Дослідження ефективності застосування обраних додаткових етапів обробки.
5. Апробація розробленого алгоритму для вирішення задачі покращення візуальної якості зображень.

# 1. АНАЛІТИЧНИЙ ОГЛЯД ЕВОЛЮЦІЙНИХ АЛГОРИТМІВ ТА ЇХ ВИКОРИСТАННЯ ПРИ ОБРОБЦІ ЗОБРАЖЕНЬ

У даній главі представлено аналітичний огляд використаної в роботі концепції еволюційних алгоритмів, яка належить до області м'яких обчислень. Розглядається структура генетичних алгоритмів та їх моделі. Наведено приклади застосування генетичних алгоритмів для обробки зображень.

## 1.1 Аналіз еволюційних алгоритмів

Еволюційні обчислення – термін, який зазвичай використовується для загального опису алгоритмів пошуку, оптимізації або навчання, що базується на формалізованих принципах природнього еволюційного процесу. Еволюційні методи призначені для пошуку кращих рішень й ґрунтуються на статистичному підході при дослідженні ситуацій та ітераційному приближенні до шуканих станів систем. На відміну від точних методів математичного програмування, еволюційні методи дозволяють знаходити рішення, що близькі до оптимальних, за прийнятний час, а також, на відміну від відомих евристичних методів оптимізації, характеризуються набагато меншою залежністю від особливостей області застосування й у багатьох випадках забезпечують кращий ступінь наближення до оптимального розв'язку. Основною перевагою еволюційних методів оптимізації є можливість вирішення багатомодальних (таких, які мають декілька локальних екстремумів) задач з великою розмірністю за рахунок поєднання елементів випадковості та детермінованості так само, як це відбувається в природному середовищі [7].

Вважається, що наукове направлення «еволюційні обчислення» є основним підходом до моделювання механізмів живої природи. Воно включає до себе три основні напрямки фундаментальних досліджень: генетичні алгоритми, еволюційне моделювання (іноді зустрічається назва «еволюційні стратегії») та еволюційне програмування [8].

Генетичні алгоритми – це нова область досліджень, яка з’явилася в результаті робіт Д. Холланда та його колег. ГА, описані Холландом, запозичили багато термінології з природної генетики. Уперше вони були застосовані до таких наукових проблем як розпізнавання образів та оптимізація. ГА являють собою адаптивний пошуковий метод, який базується на селекції кращих елементів в популяції, подібно до еволюційної теорії Ч.Дарвіна.

Генетичні алгоритми, являючи собою поєднання випадкового пошуку та точних методів, з точки зору перетворення інформації в інтелектуальних системах є перетворенням однієї скінченної нечіткої множини проміжних результатів в іншу. Важливою перевагою генетичних алгоритмів є те, що при цьому перетворенні ефективно використовується інформація, яка була накоплена в ході еволюції. Загальна схема ГА приведена на рисунку 1.1.


Рисунок 1.1 – Загальна схема ГА

При вирішенні задач оптимізації за допомогою генетичних алгоритмів кожний імовірний розв’язок кодується у вигляді *хромосоми* або *особи*, що є вектором параметрів. Усі параметри кодуються у вигляді *генів*, при чому гени можуть мати числові або функціональні значення. Числові значення беруться з певного алфавіту. Генетичний матеріал елементів зазвичай кодується на основі двійкового алфавіту {0,1}, проте, можна використовувати й буквені, а також, десяткові та інші алфавіти. На кожному кроці обробки до наявного набору хромосом (*популяції*) застосовуються *генетичні оператори*, в результаті чого популяція видозмінюється, набуваючи нових властивостей. Оцінка отриманих рішень проводиться за допомогою *фітнес-функції*, яка залежить від поставленої задачі й відповідної цій задачі цільової функції й вираховує *пристосованість* кожної хромосоми у залежності від якості відповідного їй декодованого розв’язку. Найменш пристосовані особи (особи з найгіршим значенням фітнес-функції) видаляються у процесі скорочення популяції та більше не використовуються у подальшому пошуку.

### 1.1.1 Оператори генетичного алгоритму

До класичного ГА, розробленого Д. Холландом на початку 1960-х років, входять наступні оператори:

- Селекція;
- Кросинговер;
- Мутація.

У сьогодення до цих операторів було додано декілька додаткових, які мають на меті покращити ефективність ГА. До числа додаткових операторів можна віднести, наприклад, оператор інверсії, стратегію елітаризму, стратегію інцесту тощо. На практиці додаткові оператори застосовуються доволі рідко, оскільки не завжди дають бажаний результат [2, 3], наприклад, використання стратегії елітаризму може достроково прийняти за рішення локальний екстремум.

Оператор *селекції* використовується для відбору осіб для схрещування, тобто отримання потомства. При цьому, по аналогії з природньою еволюцією, у більшості видів селекції особи з кращою адаптованістю (кращим значення фітнес-функції) мають більше можливостей для репродукції, ніж слабкі особи.

Існує багато різних методів селекції, але найбільш поширені 4 з них [9]:

- *Селекція на основі рулетки* – це найпростіший та найбільш використовуваний в генетичний алгоритмах метод. При його використанні кожному елементу в популяції відповідає зона на колесі рулетки, пропорційна до величини цільової функції. Тоді при повороті колеса рулетки кожен елемент може бути обрано для селекції з деякою ймовірністю, причому ця ймовірність тим більша, чим більше значення цільової функції елемента.

- *Селекція на основі заданої шкали* – популяція попередньо сортується від найкращої особи до найгіршої, базуючись на заданому критерії. Кожному елементу призначається певне число й далі селекція виконується відповідно до цього числа.

- *Елітна селекція* – обираються найкращі (елітні) елементи на основі порівняння цільової функції. Далі над ними виконуються різноманітні перетворення, після яких знову обираються елітні елементи. Процес виконується доти, доки продовжують з’являтися елітні елементи.

- *Турнірна селекція* – деяке число елементів, відповідно до розміру турніру, випадково або направлено обирається з популяції, й кращі елементи у цій групі на основі заданого турніру визначаються для подальшого генетичного пошуку.

Окрім описаних, існує велика кількість інших методів селекції, які можна умовно розділити на три групи [10]. До першої належать ймовірнісні. До другої – детерміновані. До третьої – різні комбінації методів першої та другої групи.

Пошуки оптимальних методів селекції й досі продовжуються, оскільки основною складністю вирішення задач оптимізації з великою кількістю локальних оптимумів є передчасна збіжність алгоритмів, тобто розв'язок потрапляє в один з не найкращих локальних оптимумів. Різноманітні методи селекції та їх модифікації якраз дозволяють у деяких випадках вирішувати проблему передчасної збіжності алгоритмів. Варто відзначити, що дослідники генетичних алгоритмів усе частіше схиляються до думки використовувати комбіновані методи селекції з використанням попередніх знань про задачі та попередні результати.

Після відбору осіб для схрещування батьківські особи обмінюються генетичним матеріалом за допомогою оператора *кросинговеру*, у результаті чого отримуються хромосоми потомків. Структура оператора кросинговеру у багатьох аспектах визначає ефективність генетичного алгоритму, тому було розроблено багато різних варіантів цього оператора [11]. Найбільш широко застосовуються 1-, 2-, n-точкові [12] та однорідний [13] оператори кросинговеру, проте, розроблено й багато інших.

Найбільш простим варіантом цього генетичного оператора є *одноточковий оператор кросинговер*. Перед початком роботи такого оператора обирається точка розрізу, як правило, випадковим чином. Після цього відбувається розріз хромосом кожного з двох батьків у цій точці та з отриманих частин шляхом перестановки утворюються два потомки.

У випадку *двоточкового оператора кросинговер* схрещування батьків проводиться подібним чином, з однією різницею, що обираються дві точки розрізу й переставляються гени, що потрапили між ними.

Одноточковий та багатоточковий оператори кросинговеру є окремими випадками *багатоточкового* або *N-точкового*, розріз генів у якому проводиться по більшій кількості точок. Проте, збільшення кількості числа точок розрізу

може мати негативний вплив на якість роботи генетичного алгоритму, оскільки може призвести до втрати «гарних» батьківських якостей.

Доволі часто застосовується *однорідний оператор кросинговеру.* У ньому замість точок розрізу використовуються двійкові маски, довжина яких співпадає з довжиною гена. Ці маски генеруються випадковим чином або обираються із попередньо визначеного набору. Кількість нулів та одиниць, як правило, приблизно рівна, проте, застосовується й *параметризований однорідний оператор кросинговеру*, у якому кількість нулів або одиниць більша. Потомки при цьому отримуються шляхом побітової операції XOR з одним із батьків.

При вирішенні задач пошуку поєднань різних елементів часто застосовується *порядковий оператор кросинговеру*. Кодування елементів перестановок у цьому випадку може реалізовуватися за допомогою буквеного алфавіту, кожен ген у цьому випадку являє собою рядок, який складається з букв цього алфавіту. Як і в одноточковому ОК, у порядковому ОК випадковим чином обирається точка розрізу. Потім лівий сегмент першого батька копіюється в ген першого потомка. Інші елементи першого потомка беруться з другого батька в упорядкованому вигляді зліва направо, виключаючи елементи, які уже потрапили з першого батька. Другий потомок формується аналогічно з лівої частини другого батька й доповнюється генетичним матеріалом першого.

Інший варіант кросинговеру, що використовується при вирішенні задач пошуку перестановок – *частково-відповідний.* У цьому випадку потомку в незміненому вигляді співставляється права частина одного з батьків. Ліва частина береться у іншого батька, проте, для виключення повторів елементів проводиться заміна повторюваних елементів на відсутні.

В оптимізаційних задачах мають місце різного типу *«жадібні» оператори кросинговеру*. Вони базуються на аналізі ЦФ рішень після кожного кроку «жадібного» ОК [8]. Він може бути реалізований на двох і більше хромосомах, максимум на всій популяції. Схема типового модифікованого алгоритму «жадібного» ОК приведена на рисунку 1.2.

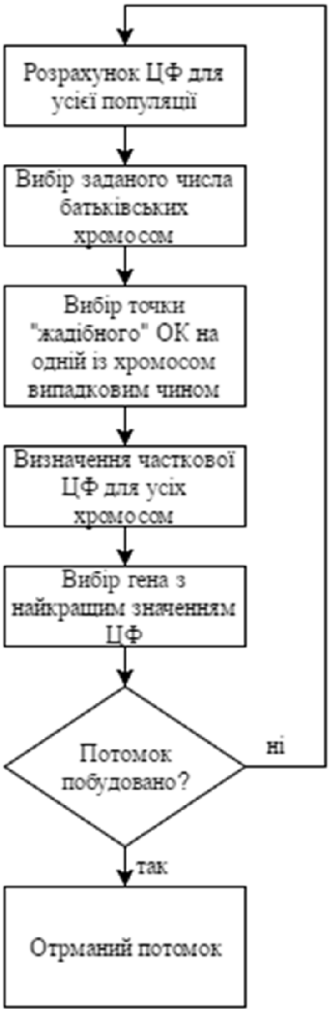


Рисунок 1.2 – модель «жадібного» ОК

На першому кроці вираховується значення цільової функції для кожної особи в популяції. Потім будь-яким чином відбувається відбір необхідної кількості батьків для схрещування та випадковим чином визначається точка «жадібного» ОК на одній із хромосом. Після цього для і-го гена кожної хромосоми, що знаходиться зліва від точки «жадібного» ОК, вираховується часткова ЦФ, тобто вартість шляху від і-го гена до гена, що знаходиться поруч.

В хромосому потомок обирається ген з найкращим значенням ЦФ. Процес побудови завершується тоді, коли потомок буде побудовано повністю.

У процесі роботи «жадібного» оператора кросинговеру можливе виникнення циклів або потрапляння у глухий кут. У такому випадку, в потомки обираються нерозглянуті гени з найкращою ЦФ.

У деяких випадках можлива збіжність генетичного алгоритму до локального екстремуму або навіть потрапляння у глухий кут. Це відбувається через знаходження усіх осіб популяції у обмеженій множині, обмін генами у якому не призводить до виходу за її межі. Для виключення таких ситуацій служить оператор *мутації*. Він випадковим чином змінює гени, що дозволяє шукати глобальний екстремум та при необхідності виходити з глухих кутів.

Існує декілька різновидів операторів мутації. Найбільш простий варіант при бінарному кодуванні генів – інверсія одного чи декількох випадково обраних бітів гену.

Другий простий варіант мутації – *одноточкова мутація*. У цьому випадку обирається випадковий біт у гені, який обмінюється місцями з бітом, що стоїть поруч. У випадку *двоточкової мутації* місцями обмінюються два випадково обрані гени.

ОМ, на відміну від ОК, є унарною операцією. При цьому ймовірність мутації значно нижча ймовірності кросинговеру.

### 1.1.2 Моделі генетичних алгоритмів

Першою була модель *Канонічних ГА*, запропонована Дж. Холландом в [14]. У даній моделі спочатку обираються розмір популяції та розрядність генів. Ці параметри залишаються незмінними протягом усього процесу еволюції. Особи для схрещування обираються випадково, й після схрещування потомки займають місця батьків. При цьому оператори кросинговеру та мутації також є найпростішими – одноточковими [15].

Другою моделлю ГА є модель *Генітор*, яку розробив Дерел Вайтлі [16, 17, 18]. Головною особливістю, що вирізняє цю модель, є оригінальна стратегія вибору осіб для схрещування. Як і в канонічному ГА, особи для схрещування обираються випадковим чином, проте на кожному кроці еволюції схрещується лише одна пара батьків, даючи при цьому лише одного потомка. Цей потомок займає у популяції місце найменш пристосованої особи. Обмежень на оператори кросинговеру та мутації немає, вони обираються на розсуд розробника.

Ця модель ГА виключає можливість утрати найбільш пристосованих осіб й показує кращу збіжність та пошук гіперплощин, ніж канонічний ГА.

Ларі Ешельманом була розроблена модель *CHC (Cross generational elitist selection, Heterogeneous recombination, Cataclysmic mutation)* [19]. У даній моделі обирається невеликий фіксований розмір популяції, який у поєднанні з однорідним оператором кросинговеру й відсутністю мутацій приводить до дуже швидкої збіжності.

Для схрещування усі особи розбиваються на пари, але схрещуються лише ті пари, для яких відстань Хеммінга більша деякого порогового (також можливі обмеження на мінімальну відстань між крайніми неоднаковими бітами). Після схрещування до наступної популяції відбираються лише найбільш пристосовані особи з потомків та батьків, причому не допускається попадання однакових осіб до наступної популяції.

Під час схрещування використовується так званий HUX-оператор (Half Uniform Crossover), різновид однорідного кросовера – кожному потомку переходить рівно половина бітів кожного батька.

Дана модель володіє доволі швидкою збіжністю, але часто сходиться до локального екстремуму. У зв’язку з цим після знаходження рішення еволюція перезавантажується за допомогою сильної мутації, що застосовується до усіх осіб, крім найбільш пристосованої.

Ще одним варіантом розвитку еволюційного підходу є *острівна модель ГА*. Ця модель була підказана природою і являє собою розвиток кількох популяцій на островах зі слабким сполученням. Популяції на кожному з островів розвиваються незалежно й можуть реалізовувати будь-які моделі ГА без обмежень на параметри та оператори. Через кілька поколінь відбувається міграція між популяціями, в результаті яких вони діляться генетичним матеріалом. При цьому відбір осіб для обміну може проходити різними способами: навмання, найбільш пристосованих осіб, може відбиратися будь-яка кількість осіб та обмін може проводитися як між усіма, так і між кількома парами островів.

Дана модель ГА має ряд важливих переваг. По-перше, усі популяції розвиваються незалежно, а отже, дана модель добре підходить для паралельних обчислень. При цьому кожна популяція має відносно невеликий розмір, а тому, обробка кожного покоління буде займати відносно небагато часу. По-друге, на кожному з островів можна використовувати різні стратегії відбору, оператори кросинговеру й мутації, що підвищує універсальність алгоритму. Крім того, наявність декількох незалежних популяцій розширює область пошуку навіть з невеликою ймовірністю мутації, оскільки знижується ймовірність потрапляння усієї популяції в область одного локального екстремуму.

Окрім перерахованих, існує *Гібридна модель ГА,* яка визначає не стільки вид самого ГА, скільки стратегію його застосування при вирішенні задачі [15]. ГА є робастним алгоритмом, тобто дозволяє швидко знайти рішення близьке до оптимального, але може потребувати багато часу для пошуку оптимального. У зв’язку з цим у деяких підходах ГА застосовується лише на початкових етапах для звуження області пошуку рішень й попадання в глобальний мінімум, а потім проводиться пошук рішень за допомогою класичних детермінованих методів.

Такий підхід дозволяє поєднувати сильні сторони генетичних алгоритмів й детермінованих методів оптимізації. Крім того, можна використовувати «класичні» методи, як-от наприклад, hill-climbing, й усередині самих ГА. На

кожному поколінні кожний отриманий потомок оптимізується цим методом таким чином, що кожна особа досягне локального максимуму, біля якого вона знаходиться, після цього відбувається відбір, схрещування і мутація. Такий метод погіршує здатність алгоритму шукати рішення за допомогою відбору гіперплощин, зате зростає ймовірність того, що одна з осіб потрапить у область глобального максимуму й після оптимізації опиниться поруч з розв'язком задачі.

## 1.2 Розробка алгоритму обробки зображень, використовуючи ГА

На сьогодні цифрові зображення є невід'ємною частиною багатьох сфер людської діяльності та їх вплив неупинно росте. Це пояснюється відносною простотою їх отримання, наочністю та зручністю представлення інформації у подібній формі, легкістю копіювання та поширення й багатьма іншими факторами. Застосування цифрових зображень в науці дало людству новий, дуже ефективний інструмент для проведення досліджень. Використання електронних зображень у техніці дозволило відкрити нові області робіт. Дистанційне зондування, біомедичний аналіз несправностей, контроль якості, дистанційне керування об'єктами, системи безпеки – усі вони у тій чи іншій формі використовують цифрові зображення.

Проте, у всіх цих областях крім безпосереднього використання, часто потрібна автоматична обробка зображень. Дуже часто постають завдання сегментації зображень, автоматичного розпізнавання об'єктів, підвищення якості, виділення специфічних характеристик, пошуку об'єктів та багато інших. Однак більшість завдань обробки зображень, з якими легко справляється людський мозок, дуже важко піддаються формалізації і адаптації в машинній обробці. Це обумовлено багатьма причинами, серед яких можна виділити наступні:

- Надзвичайна складність людського мозку. Зорова система людини є багаторівневою вкрай високорозвиненою системою сприйняття та

розпізнавання образів. Накопичений досвід дозволяє людині не тільки виділяти і ідентифікувати об’єкти, але і доповнювати їх, оцінювати їх в контексті всієї сцени і виконувати інші операції. Наприклад, в абсолютній більшості випадків, людина легко відрізнить зображення будь-якого об’єкта від нього самого, проте з точки зору машинного уявлення вони практично не відрізняються.

- Однорідність і обмеженість машинного представлення зображень. Кожна точка цифрового зображення являє собою набір деяких характеристик, які залежать від використовуваного колірного простору. Людина ж отримує значно більше інформації і легше інтерпретує її. Людина може оцінювати відстань до об'єктів і їх розміри за рахунок стереоскопічного зору, враховує інформацію про освітлення та інші параметри сцени.

- Складність оцінки результату обробки. У більшості загальних завдань обробки зображень автоматична оцінка якості обробки сильно ускладнена, тому що їх рішення часто суб’єктивне. Крім того, вкрай складно оцінити контекст зображення, умови його отримання, мету обробки тощо. Ці проблеми частково вирішуються для вузькоспеціалізованих завдань, проте і в них оцінка результату є недостатньо точною.

Проте, для більшості завдань обробки зображень є різні рішення. При цьому для обробки зображень широко застосовуються генетичні алгоритми, тому що вони дозволяють проводити обробку навіть при відсутності формального опису проблеми завдяки можливості накопичення інформації в процесі еволюції і гарній пристосованості для вирішення багатопараметричних задач.

### 1.2.1. Покращення зображень

Практично у всіх областях використання цифрових зображень висовують дуже високі вимоги до їх якості. Для успішного застосування, зображення повинно мати високий рівень контрасту, хорошу яскравість, низький рівень шумів тощо. При цьому, не дивлячись на високий рівень розвитку методів і засобів отримання зображень, виконання даних вимог не завжди можливе. Умови середовища, в якому отримують зображення, часто роблять їх якість практично неприйнятною. Виходом з такої ситуації може служити застосування методів підвищення якості зображень.

Метод підвищення якості зображень – сукупність процедур, що виконуються над цифровим зображенням для підвищення його візуальних якостей та позбавлення від недоліків, таких як шуми, низький контраст, недостатня яскравість тощо.

#### 1.2.1.1. Покращення зображень генетичними алгоритмами

Одним з підходів до покращення зображень є застосування генетичних алгоритмів.

У загальному випадку їх застосування базується на припущенні про існування деякої функції перетворення яскравості точки, що дозволяє поліпшити деякі характеристики зображення. Для підвищення адаптивності перетворення в даній функції, як правило, вводяться залежності від деяких параметрів зображення, таких як середнє значення яскравості по зображенню, середнє значення яскравості в деякому околі точки, дисперсія в цьому околі тощо. Крім того, зважаючи на складність вибору типу функції в ній, як правило, присутня деяка кількість незалежних параметрів, що дозволяють впливати на характеристики перетворення [20, 21, 22, 23]. Прикладом такого перетворення може бути функція, запропонована в [23]:

$$L^*(x,y) = T(L(x,y)) = \left(k\frac{M}{\sigma(x,y)+b}\right)\left(L(x,y) - cm(x,y)\right) + m(x,y)^a, \qquad (1)$$

де $L^*(x,y)$ і $L(x,y)$ – отримана та початкова яскравість точки з координатами $(x,y)$, $M$ – глобальний середній показник яскравості зображення, $m(x,y)$ – середнє значення яскравості в околі точки з відповідними координатами, $\sigma(x,y)$ – значення середньоквадратичного відхилення в цьому ж околі; $a,b,c,k$ – невідомі параметри перетворення.

Для вирішення завдання покращення зображення досить підібрати оптимальні параметри перетворення, а потім застосувати його до кожної точки зображення. З огляду на те, що в даному випадку має місце багатопараметрична задача оптимізації, одним з варіантів її вирішення є застосування ГА.

Хромосоми генетичного алгоритму в даному випадку кодують параметри перетворення. Тобто в наведеному прикладі кожна хромосома містить чотири гени, що є числами з плаваючою точкою, кожен з генів відповідає одному параметру перетворення. Підбір операторів генетичного алгоритму виробляється на розсуд дослідника, проте при цьому не можна забувати, що розмір зображення може бути досить великим, а тому оператори повинні не тільки забезпечувати якісний пошук рішень, а й мати не надто високу обчислювальну складність.

Важливою проблемою при такому підході покращення зображень є вибір цільової функції генетичного алгоритму. Складність підбору цільової функції пов’язана з відсутністю загальноприйнятого критерію якості зображень. Якість зображення багато в чому є суб’єктивним параметром, і навіть при експертній оцінці можна отримати різні результати від різних експертів. Підібрати ж стовідсотково точний об’єктивний критерій взагалі не є можливим. Проте, використовуючи поєднання різних існуючих критеріїв можна домогтися досить точної оцінки.

Одним з параметрів, які визначають якість зображень, є контраст. Оскільки зображення має складний сюжетний характер, то це породжує необхідність при визначенні його контрастності виходити з контрасту окремих комбінацій елементів зображення. При цьому всі елементи вважаються рівнозначними, і контраст кожної їх пари обчислюється за формулою

$$C_{ij} = \frac{L_i - L_j}{L_i + L_j}, \quad (2)$$

де $L_i, L_j$ – яскравість елементів сюжетного зображення.

Сюжетність зображення передбачає можливість його використання людиною. Тому, при оцінці контрасту як одного з параметрів якості зображення, необхідно враховувати ряд особливостей зорового сприйняття людини.

Далі, застосовуючи правило підсумовування контрастів, обчислюють набір величин, які визначають сприйняття кожної пари елементів зображення. Проводячи усереднення матриці локальних контрастів, отримують сумарний контраст. Отриманий результат може бути використаний як один з параметрів оцінки візуальної якості зображення [23].

Контрастність зображення, як і його різкість можна оцінити за кількістю та яскравістю крайових пікселів. Крайові пікселі – це точки зображення, що лежать на межі поділу областей з різною яскравістю. Чим більше таких точок на зображенні, тим вища його різкість. А яскравість таких точок показує рівень контрасту.

Для виявлення крайових пікселів використовуються крайові детектори. Існує безліч підходів до пошуку крайових пікселів, але практично всі вони належать до двох категорій:

- Методи, засновані на пошуку максимумів.
- Методи, засновані на пошуку нулів.

Методи, засновані на пошуку максимумів, виділяють границю за допомогою обчислення «сили краю», зазвичай вираз першої похідної, такий як величина градієнта, і потім пошуку локальних максимумів сили краю, використовуючи передбачуваний напрямок границі, зазвичай це перпендикуляр до вектору градієнта. Методи, засновані на пошуку нулів, шукають перетин осі абсцис виразу другої похідної, у вигляді нулів Лапласіана або нулів нелінійного диференціального виразу. В якості кроку підготовки до виділення границь практично завжди застосовується згладжування зображення, найчастіше фільтром Гауса. Найбільш широко застосовуються такі детектори краю, як оператор Кенні, оператор Собеля, оператор Прюітт і перехресний оператор Робертса.

Існує ще один метод оцінки якості зображення. Його суть полягає в наступному. Експериментально було встановлено, що оптимальне, з точки зору суб'єктивного сприйняття, зображення має нормальний розподіл яскравості його елементів. Для зручності у подальших розрахунках було застосовано критерій нормального розподілу. За ступенем відхилення реального розподілу яскравості від нормального проводилася оцінка якості зображення. Крім кількісної оцінки якості зображення, даний метод дозволяє отримати інформацію про наявність та вагове співвідношення градацій яскравості зображення. Результати оцінки якості зображення, отримані за цим методом, добре корелюють з суб'єктивною оцінкою візуальної якості зображення [24].

Ще один параметр, що відображає легкість сприйняття зображення людиною – рівень адаптації за яскравістю зорової системи людини $\overline{L}$, оптимальним значенням якої є половина максимально можливого діапазону яскравості $L_{max}/2$ . Тому величину відхилення $\bar{L}$ від $L_{max}/2$ можна використовувати як оцінку рівня адаптації зорової системи:

$$LQ = 1 - \frac{\bar{L} - L_{max}/2}{L_{max}/2}. \qquad (3)$$

Іншим важливим параметром оцінки візуальної якості зображення є повнота використання його елементами градацій яскравості. Аналітичне вираження цього параметра таке:

$$KQ = \frac{S}{L_{max}}, \tag{4}$$

де S – кількість рівнів яскравості, для кожного з яких на даному зображенні є більше, ніж $\delta$×M×N кількість елементів з даною яскравістю (N та M – розміри зображення, $\delta$ – деяка константа).

Однією із варіацій даного критерію є ентропія зображення. Вона обчислюється на основі гістограми зображення наступним чином.

Спочатку будується гістограма зображення, кількість рівнів якої відповідає кількості градацій яскравості. Граничні значення рівня з індексом $i$ записуються як $A_i$ та $B_i$ для $i \in \{0, \dots, 255\}$. На основі гістограми вводиться міра ентропії зображення $I$, як

$$H(I) = \sum_i v(i);$$
$$v(i) = \begin{cases} -\nu_i \log_2(\nu_i), \nu_i \neq 0 \\ 0, \nu_i = 0 \end{cases}, \tag{5}$$

де $\nu_i$ – частота пікселів, що мають рівні градацій яскравості між граничними значеннями $A_i$ та $B_i$. Відзначимо, що така міра не є реальною ентропією, оскільки $\nu_i$ – скоріше відносні частоти, ніж ймовірності.

У результаті застосування генетичного алгоритму для підбору параметрів перетворення $T$ вдається отримати значне покращення якості зображення, проте, даний метод має суттєвий недолік – високу обчислювальну складність та пов’язану з цим низьку швидкість обробки.

## 1.3 Основні результати та висновки з розділу 1

1. Проведено аналіз еволюційного підходу вирішення задач оптимізації. Розглянуто структуру генетичного алгоритму, його складові частини.
2. Проведено аналіз існуючих моделей генетичних алгоритмів, показана необхідність вибору моделі, яка найбільше підходить для вирішення задачі обробки зображень.
3. Розглянуто приклади застосування генетичних алгоритмів для вирішення задач обробки зображень. Проаналізовано достоїнства такого підходу, а також складнощі, що виникають під час еволюційної обробки.

# 2. ЗАСТОСУВАННЯ ГЕНЕТИЧНОГО АЛГОРИТМУ ДЛЯ ПОКРАЩЕННЯ ЗОБРАЖЕНЬ

У даній главі наводиться опис розробленого генетичного алгоритму для підвищення якості зображень: загальна схема застосування, використані функції, методи оцінки зображень та інші аспекти. Наводяться результати оцінки ефективності застосування різних моделей ГА для вирішення даної задачі. Показано результати тестування розробленого алгоритму на наборі тестових зображень та оцінка параметрів роботи алгоритму.

## 2.1 Схема застосування генетичного алгоритму

Загальна схема застосування ГА для покращення зображень наведена в Главі 1. Розглянемо її детальніше.

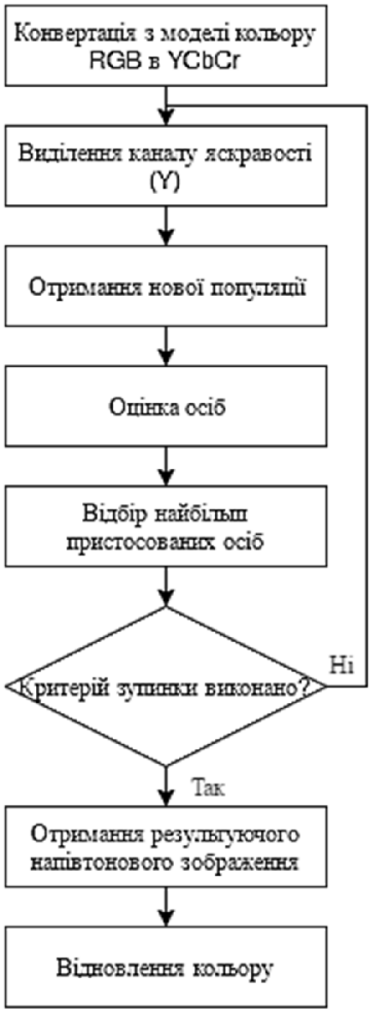


Рисунок 2.1 – Алгоритм покращення зображень на основі ГА

Як видно зі схеми алгоритму, наведеної на рисунку 2.1, застосування генетичного алгоритму для поліпшення зображень включає як специфічні етапи обробки, так і етапи, загальні для всіх генетичних алгоритмів. На першому етапі проводиться виділення каналу яскравості зображення, який далі піддається обробці. При поліпшенні зображень, як правило, проводиться зміна саме яскравості точок, тому що колірна інформація значно менше впливає на візуальне сприйняття, ніж яскравість. Виділення складової яскравості може проводитися різними способами. У даній роботі був обраний перехід від колірної моделі RGB до моделі YCbCr, в якій компонента Y є поданням яскравості, а компоненти Cb і Cr відображають кольорову інформацію, яка може бути відкинута під час обробки.

На наступному етапі ініціалізується та запускається безпосередньо генетичний алгоритм. Він може бути реалізований за будь-якою з існуючих моделей, вибір яких проводиться дослідником на підставі емпіричних даних або специфіки завдання.

Кожна особа при даному застосуванні ГА зберігає інформацію про перетворення зображення, яке буде отримано з її використанням. Таким чином, популяції відповідає набір різних варіантів перетворення вихідного зображення, найкращий з яких обирається в процесі еволюції.

Найбільш специфічним етапом є етап оцінки осіб. При оцінці особи до вихідного зображення застосовується перетворення, яке визначається цією особою. Потім отримане зображення оцінюється за допомогою обраної функції оцінки якості зображення. Відповідно до цих оцінок проводиться відбір найбільш пристосованих осіб, як правило, за допомогою простого порівняння.

На жаль, при вирішенні даної задачі відсутнє еталонне зображення. Це призводить до того, що функції оцінки зображень не мають ідеального значення – вони унікальні для кожного зображення та максимізуються або мінімізуються. Це не дозволяє зупиняти роботу алгоритму при знаходженні рішення із заданою

точністю, і доводиться застосовувати інші критерії зупинки, такі як обмеження на кількість поколінь або відсутність значних змін в популяції протягом досить тривалої кількості поколінь.

Після виконання критерію зупинки обирається особа з найкращим значенням функції пристосованості і на основі її генів отримуємо таблицю підсумкових яскравостей точок. Потім проводиться відновлення колірної інформації. Для цього застосовується або зворотний перехід від колірної моделі YCbCr до моделі RGB, або пропорційна зміна значення кожної з колірних компонент вихідного зображення. У даній роботі була обрана пропорційна зміна колірних компонент кожної точки зображення відповідно до зміни яскравості цієї точки. Даний метод призводить до менших спотворень кольору, ніж інші методи.

## 2.2 Функції, що використовуються при обробці

У даному розділі будуть розглянуті основні функції, що застосовуються при роботі ГА. Ці функції використовуються при перетворенні зображень та оцінці їх якості.

### 2.2.1 Ядро покращення зображень

Основою розроблюваного алгоритму є ядро покращення зображень.

Ядро покращення - це перетворення T, що застосовується до кожної точки зображення з координатами (x, y). Вхідним параметром для цього перетворення є початкова яскравість точки f (x, y), значення на виході – остаточна яскравість точки g (x, y):

$$g(x,y) = T\big(f(x,y)\big) \tag{6}$$

При цьому для досягнення хороших результатів обробки та підвищення адаптивності алгоритму необхідно, щоб це перетворення залежало не тільки від яскравості самої точки, до якої воно застосовується, а й від яскравості точок в

деякому околі чи на усьому зображенні. У [23] перетворення $T$ визначено наступним чином:

$$g(x,y) = T\big(f(x,y)\big) = \left(k\frac{M}{\sigma(x,y)+b}\right)\big(f(x,y) - cm(x,y)\big) + m(x,y)^a; \quad \begin{matrix} x = \overline{0 \ldots H_{size}-1}; \\ y = \overline{0 \ldots W_{size}-1}; \end{matrix} \tag{7}$$

де $g(x,y)$ і $f(x,y)$ – отримана та початкова яскравість точки з координатами $(x,y)$, $M$ – глобальний середній показник яскравості зображення, $m(x,y)$ – середнє значення яскравості в околі точки з відповідними координатами, $\sigma(x,y)$ – значення середньоквадратичного відхилення в цьому ж околі, $H_{size}$ та $W_{size}$ – висота та ширина зображення відповідно.$a, b, c, k$ – параметри ядра покращення, що визначаються за допомогою генетичного алгоритму.

В ході проведення досліджень було виявлено, що застосування середньоквадратичного відхилення в чистому вигляді призводить до не дуже гарних результатів, оскільки в однотонних областях його значення близьке до нуля, що призводить до сильного освітлення. Крім того, в областях з різким перепадом яскравості (наприклад, на границях об’єктів) спостерігається досить сильна зміна яскравості, що приводить до появи ореолів у великих об’єктів, що контрастують з фоном. Тому, середньоквадратичне відхилення прибираємо з формули узагалі.

### 2.2.2 Критерій якості зображення

Фітнес функція є однією з найважливіших частин генетичного алгоритму. Саме від неї залежить, які особи будуть відібрані для продовження роботи алгоритму. Тому необхідно вибрати такі оцінки, які найбільш точно відображають якість оброблюваного зображення.

Проте, якість зображень – суб’єктивне поняття, і навіть при експертній оцінці можна отримати різні результати від різних експертів. Підібрати ж стовідсотково точний об’єктивний критерій взагалі не представляється

можливим. Однак, використовуючи поєднання різних існуючих критеріїв можна домогтися досить точної оцінки. Розглянемо їх.

Якість зображення визначається великою кількістю технічних характеристик системи: співвідношенням сигнал/шум і статистичними характеристиками шуму, градаційними характеристиками, спектральними (колірними) характеристиками, інтервалами дискретизації і т.д.

Досить широко застосовується комплексна оцінка якості зображень такого вигляду [23]:

$$F(x) = \ln(\ln(E(I)) + e)\frac{\eta(I)}{H_{size}W_{size}}e^{H(I)}, \quad (8)$$

де $\eta(I)$ – кількість крайових пікселів, $E(I)$ – їх сумарна інтенсивність, $H(I)$ – ентропія зображення.

Розглянемо кожну змінну більш детально. Почнемо з кількості крайових пікселів та їх сумарної інтенсивності.

Контрастність зображення, а також його різкість можна оцінити за кількістю і яскравістю крайових пікселів. Крайові пікселі – це точки зображення, що лежать на межі областей з різною яскравістю. Чим більше таких точок на зображенні, тим вища його різкість. А яскравість таких точок показує рівень контрасту.

Для виявлення крайових пікселів використовуються крайові детектори. Один з таких детекторів - оператор Собеля.

Оператор Собеля [25, 26, 27] проводить вимірювання двовимірного просторового градієнта на зображенні і виявляє області з великим значенням цього параметра. Ці області і відповідають краям. Як правило, він використовується для оцінки модуля градієнта в кожній точці чорно-білого зображення.

Робота цього оператора базується на накладанні на кожну точку зображення двох масок обертання. Ці маски є двома ортогональними матрицями розмірністю 3x3 (рис. 2.2). Вони виявляють границі, розташовані вертикально і горизонтально на зображенні. При роздільному накладенні цих масок на вихідне зображення можна отримати оцінку градієнта по кожному з напрямків (позначимо їх $G_x$ і $G_y$). Кінцеве значення градієнта отримуємо по формулі:

$$G(x,y) = \sqrt{{G_x}^2(x,y) + {G_y}^2(x,y)}. \tag{9}$$

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

| | | |
|---|---|---|
| +1 | +2 | +1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

Рисунок 2.2 – матриця
оператора Собеля

За допомогою даного оператора вираховується кількість крайових пікселів $\eta(I)$ та їх сумарна інтенсивність $E(I)$:

$$E(I) = \sum_x \sum_y G(x,y);$$
$$\eta(I) = \sum_{x,y} \mu(x,y); \tag{10}$$
$$\mu(x,y) = \begin{cases} 1, G(x,y) > a \\ 0, G(x,y) < a' \end{cases}$$

де *a* – деяке граничне значення. На рисунку 2.3. наведений приклад зображення та результат обробки його оператором Собеля. Світлі точки відповідають крайовим пікселям. Причому, чим світліша точка – тим вища інтенсивність крайового пікселя.

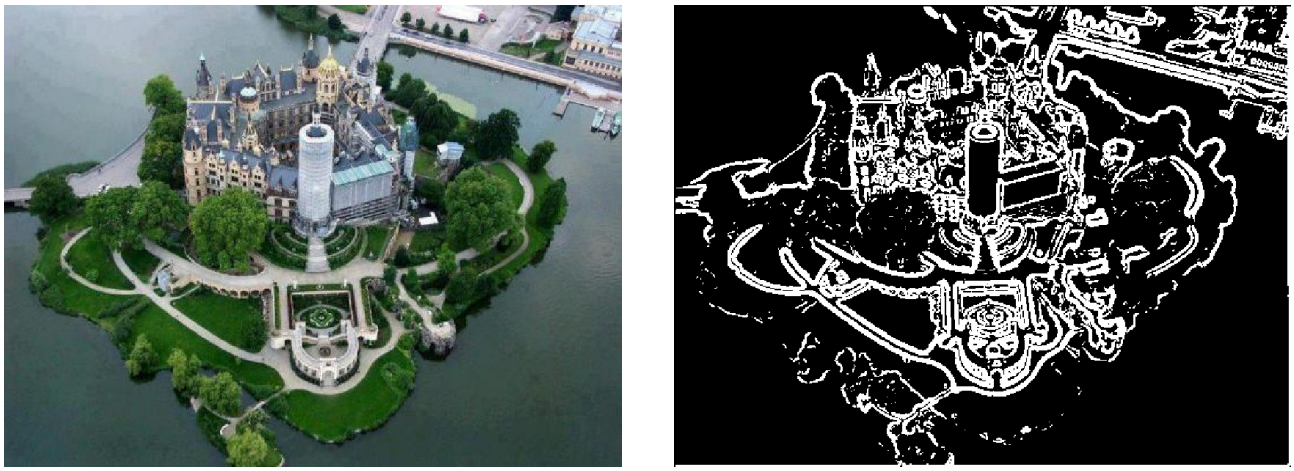

Рисунок 2.3 – Оригінальне зображення (зліва), оброблене оператором Собеля зображення (справа) [37]

Тепер розглянемо ентропію зображення. Вона вираховується на основі гістограми зображення таким чином.

Спочатку будується гістограма зображення, кількість рівнів якої відповідає кількості градацій яскравості. Граничні значення рівня з індексом $i$ записуються як $A_i$ та $B_i$ для $i \in \{0, \dots, 255\}$. На основі гістограми вводиться міра ентропії зображення $I$, як

$$H(I) = \sum_i v(i);$$
$$v(i) = \begin{cases} -\nu_i \log_2(\nu_i), \nu_i \neq 0 \\ 0, \nu_i = 0 \end{cases}, \quad (11)$$

де $\nu_i$ – частота пікселів, що мають рівні градацій яскравості між граничними значеннями $A_i$ та $B_i$. Відзначимо, що така міра не є реальною ентропією, оскільки $\nu_i$ – скоріше відносні частоти, ніж ймовірності.

Запропонована оцінка якості зображення досить добре корелює з експертними оцінками, і її було вирішено взяти за основу. Однак з урахуванням специфіки методу її було вирішено доповнити ще двома параметрами.

Оскільки значення яскравості мають розмірність байта, то при перетворенні зображення з певними параметрами часто виникають перевищення

допустимих значень яскравості, що призводить до втрати інформації. Для запобігання таких перевищень необхідно додати облік кількості переповнень в оцінку якості зображення:

$$OC = 1 - \frac{N_{ovf}}{H_{size}W_{size}}, \tag{12}$$

де $N_{ovf}$ – кількість точок з яскравістю вище граничної, що з'явилися при обробці. Цей параметр є скоріше характеристикою алгоритму, ніж зображення, проте введення його в оцінку якості дозволяє значно знизити пристосованість осіб, параметри яких призводять до втрат інформації при обробці.

Крім того, середня яскравість по зображенню в результаті обробки може знизитися або підвищитися, і при цьому можливо отримати занадто темне або занадто світле з точки зору людини зображення. Для підвищення пристосованості осіб, що мають зображення з оптимальною середньою яскравістю, в оцінку якості зображення було введено рівень адаптації зору людини до зображення, базуючись на яскравості. Найбільш приємні для людського ока зображення є ті, середня яскравість яких лежить в середині діапазону яскравості. Рівень адаптації виражається у вигляді такої формули:

$$LQ = 1 - \left| \frac{\overline{L} - \frac{L_{max}}{2}}{\frac{L_{max}}{2}} \right|, \tag{13}$$

де $\overline{L}$ – середня яскравість зображення, а $L_{max}$ – максимальне значення яскравості.

Кінцева формула для оцінки якості зображення має наступний вигляд:

$$F(x) = \ln(\ln(E(I)) + e)\frac{\eta(I)}{H_{size}W_{size}}e^{H(I)} \cdot OC \cdot LQ \tag{14}$$

## 2.3 Варіанти реалізації ЯЕО в контексті обробки зображень

### 2.3.1 Розробка ядра еволюційних обчислень

В результаті вивчення і розвитку еволюційних обчислень різними вченими, був розроблений ряд моделей генетичних алгоритмів.

Всі ці моделі засновані на ідеї еволюції, проте реалізація процесу еволюції в кожній з них різна. Завдяки застосуванню різних генетичних операторів або змін в процесі генерації нових поколінь популяції вдається підвищити збіжність алгоритму або скоротити час його роботи.

Розв’язуване в даній роботі завдання має значну обчислювальну складність. Найбільш складною в обчислювальному плані є процедура оцінки пристосованості особи, а значить, необхідно вибрати таку модель генетичного алгоритму, яка вимагає найменшу кількість таких оцінок.

Крім того, алгоритм повинен мати гарну збіжність і швидко знаходити «хороше» рішення в задачі максимізації функцій. Це означає, що алгоритм повинен мати досить широкий простір пошуку, проте здатність до використання знайдених рішень може бути відносно слабкою. Виходячи з цього, можна вибрати досить високу ймовірність мутації і оператори кросинговеру з високою руйнівною здатністю.

Крім того, необхідно щоб витрати часу на роботу операторів самого алгоритму теж були мінімальні.

Перераховані вище характеристики можна об'єднати в два основних критерії, які і будуть порівнюватися для всіх моделей:

- час обробки;
- оцінка найбільш пристосованої особи після отримання N потомків.

Для реалізації та порівняння були обрані 3 моделі ГА:

1. Канонічний ГА;
2. Генітор;
3. CHC.

У роботі не реалізовувалися і не тестувалися гібридні і паралельні моделі ГА, оскільки досліджувана функція сходиться досить швидко і знаходження її рішень іншими детермінованими або стохастичними методами має ще більшу обчислювальну складність, ніж пошук рішень за допомогою ГА. У зв'язку з цим паралельні і гібридні моделі будуть програвати класичним по швидкості, а якість обробки залишиться на тому ж рівні.

Крім того, слід зазначити, що всі обрані для дослідження моделі реалізуються з фіксованими параметрами, такими як розмір популяції і ймовірність мутації, в той час як ряд дослідників пропонує застосовувати адаптивне підстроювання параметрів ГА [28, 29, 30, 31, 32, 33]. Адаптивні моделі в даній роботі не розглядаються тому, що передбачуване число поколінь еволюції невелике і адаптивний підбір параметрів не дасть значної переваги, а навпаки призведе до ускладнення алгоритму.

### 2.3.2 Опис реалізованих моделей генетичних алгоритмів

#### 2.3.2.1 Канонічний ГА

Канонічний ГА є найбільш простою моделлю еволюції. Це класичний генетичний алгоритм, запропонований Дж. Холландом. У роботі цього алгоритму застосовуються найпростіші оператори:

• одноточковий кросинговер;

• одноточкова мутація;

• випадковий відбір осіб для схрещування.

Розмір популяції і розрядність генів фіксовані, елітизм відсутній. Такі параметри алгоритму спрощують його розуміння, проте його збіжність, як правило, нижча, ніж збіжність алгоритмів інших моделей.

Передбачається, що витрати часу на роботу операторів такого генетичного алгоритму будуть невеликі, і швидкість обробки буде вища, ніж у алгоритму Генітор. Однак збіжність алгоритму, що базується на такій моделі, у контексті даної задачі буде відносно низькою, тому що одноточковий кросинговер має низьку руйнівну здатність, що зменшує простір пошуку.

#### 2.3.2.2 Генітор

Головною відмінністю даної моделі від канонічного ГА є специфічна стратегія відбору. Для схрещування в кожному поколінні обираються тільки дві особи, які дають тільки одного нащадка. Цей нащадок оцінюється і займає місце найменш пристосованої особи. Таким чином, на кожному кроці в популяції оновлюється тільки одна особа.

Крім того, на відміну від канонічного ГА, дана модель не накладає обмежень на тип оператора кросинговеру, а тому, з огляду на специфіку завдання, можна використовувати цей оператор з великою руйнівною здатністю. У даній роботі буде розглядатися модель Генітор, що використовує однорідний оператор кросинговеру.

Варто зазначити, що витрати часу на роботу операторів ГА в даному випадку трохи більша, ніж у випадку канонічного ГА. Це обумовлено тим, що відбір батьків відбувається для кожного окремого нащадка, а не для їх пари. Крім того, однорідний оператор кросинговеру має більшу обчислювальну складність, ніж одноточковий, але краще підходить для вирішення поставленого завдання.

#### 2.3.2.3 CHC

Абревіатура CHC розшифровується як Cross-population selection, Heterogeneous recombination and Cataclysmic mutation. Особливістю даного алгоритму є малий розмір популяції і відсутність мутації при отриманні нащадків, що призводить до швидкої збіжності. Проте, через вузький початковий простір пошуку, знайдене рішення як правило не є кращим на всьому просторі рішень. В силу цього після знаходження деякого рішення алгоритм перезавантажується, причому найкраща особа копіюється в нову популяцію, а особи, що залишилися, піддаються сильній мутації (мутує приблизно третина бітів в хромосомі), після чого пошук повторюється.

Ще однією специфічною рисою є стратегія схрещування: всі особини розбиваються на пари, причому схрещуються лише ті пари, в яких хромосоми осіб істотно різні (відстань Хеммінга більше деякого порогового плюс можливі обмеження на мінімальну відстань між крайніми різними бітами). При схрещуванні використовується так званий HUX-оператор (Half Uniform Crossover) - це різновид однорідного кросинговеру, але в ньому до кожного нащадка потрапляє рівно половина бітів хромосоми від кожного з батьків.

Безсумнівним плюсом даної моделі є швидка збіжність у вихідному просторі пошуку, але через відсутність мутацій та малий розмір популяції сам цей простір невеликий. Для пошуку найкращого рішення на всьому просторі можливих рішень потрібно кілька перезапусків, при цьому найкраще зі знайдених рішень завжди зберігається, проте кожен додатковий перезапуск збільшує час роботи.

Алгоритм на основі даної моделі працює швидше за інші на початкових етапах пошуку рішень, але при пошуку більш точного рішення витрати часу на його виконання значно збільшуються.

### 2.3.3 План тестування ядра ГА

Для тестування реалізацій розроблюваного алгоритму, що базуються на різних моделях ГА був узятий набір з 15 тестових зображень розміром від 0,09 до 0,25 мегапікселя. Відносно невеликий розмір зображень було обрано для зменшення часу обробки.

Кожне з зображень було оброблено кілька разів за допомогою кожної з реалізацій розробляється алгоритму, було збережено середній час обробки і значення функції пристосованості результуючої особи. Параметри обробки були обрані з урахуванням того, щоб кількість одержуваних в процесі еволюції осіб було однаковим для кожної з моделей. Для дотримання цієї умови і отримання кожною з моделей 100 нових осіб в процесі еволюції було обрано такі параметри:

1. Канонічний ГА. Оскільки на кожному етапі розмір популяції подвоюється, то в процесі еволюції виходить N*k осіб, де N – розмір популяції, а k – кількість поколінь. Для отримання 100 осіб було обрано популяцію в 20 осіб, а тривалість еволюції – 5 поколінь.

2. Генітор. При роботі ГА, побудованого за моделлю Генітор, на кожному поколінні з’являється рівно одна нова особа, а значить кількість поколінь має дорівнювати 100. Розмір початкової популяції був обраний таким же, як при канонічній реалізації ГА – 20 осіб.

3. СНС. Кількість нових осіб при кожному запуску еволюції в моделі СНС розраховується за тією ж формулою, що і для канонічного ГА, проте при обробці відбувається кілька перезапусків еволюції, а розмір популяції обирається меншим. Для отримання 100 осіб було вирішено проводити 2 запуски еволюції, кожен з яких складається з 5 поколінь. Популяція складається з 10 осіб.

Результати обробки наведені в таблиці 2.1, а також, на рисунках 2.4 і 2.5 наведено графічне представлення часу обробки в секундах і оцінки отриманого рішення.

Після отримання результатів для кожної оцінки була обчислена відносна різниця цієї оцінки з найкращим показником серед моделей, ця різниця є рангом оцінки. Для кожної моделі були отримані сумарні ранги за часом і якістю обробки, а загальний ранг моделі визначається сумою цих рангів. Ранги всіх оцінок і моделей наведені в таблиці 2.2.

Таблиця 2.1. Результати обробки зображень

| Номер зображення | Канонічний ГА | | Генітор | | СНС | |
|---|---|---|---|---|---|---|
| | Час | Макс. значення оцінки | Час | Макс. значення оцінки | Час | Макс. значення оцінки |
| 1 | 7,38 | 13,16 | 7,43 | 20,54 | 7,05 | 16,18 |
| 2 | 7,92 | 4,14 | 7,73 | 4,30 | 7,48 | 2,84 |
| 3 | 7,77 | 107,59 | 7,83 | 114,98 | 7,48 | 68,29 |
| 4 | 6,98 | 430,97 | 7,04 | 520,41 | 6,73 | 456,18 |
| 5 | 11,30 | 328,47 | 11,40 | 427,56 | 10,89 | 413,63 |
| 6 | 7,18 | 1,36 | 7,32 | 1,50 | 6,88 | 1,62 |
| 7 | 7,18 | 458,53 | 7,28 | 569,83 | 6,92 | 427,41 |
| 8 | 7,10 | 206,46 | 7,24 | 181,83 | 6,87 | 221,51 |
| 9 | 8,11 | 871,04 | 8,16 | 891,78 | 7,79 | 817,95 |
| 10 | 8,58 | 360,77 | 8,58 | 304,55 | 8,24 | 253,94 |
| 11 | 4,61 | 4779,13 | 4,66 | 6253,93 | 4,49 | 3664,44 |
| 12 | 11,16 | 4740,45 | 11,39 | 4835,18 | 10,74 | 4246,33 |
| 13 | 8,29 | 63,57 | 8,35 | 62,97 | 7,98 | 38,23 |
| 14 | 7,11 | 14,41 | 7,18 | 11,77 | 6,92 | 18,33 |
| 15 | 3,52 | 1385,19 | 3,52 | 1446,48 | 3,37 | 1478,02 |

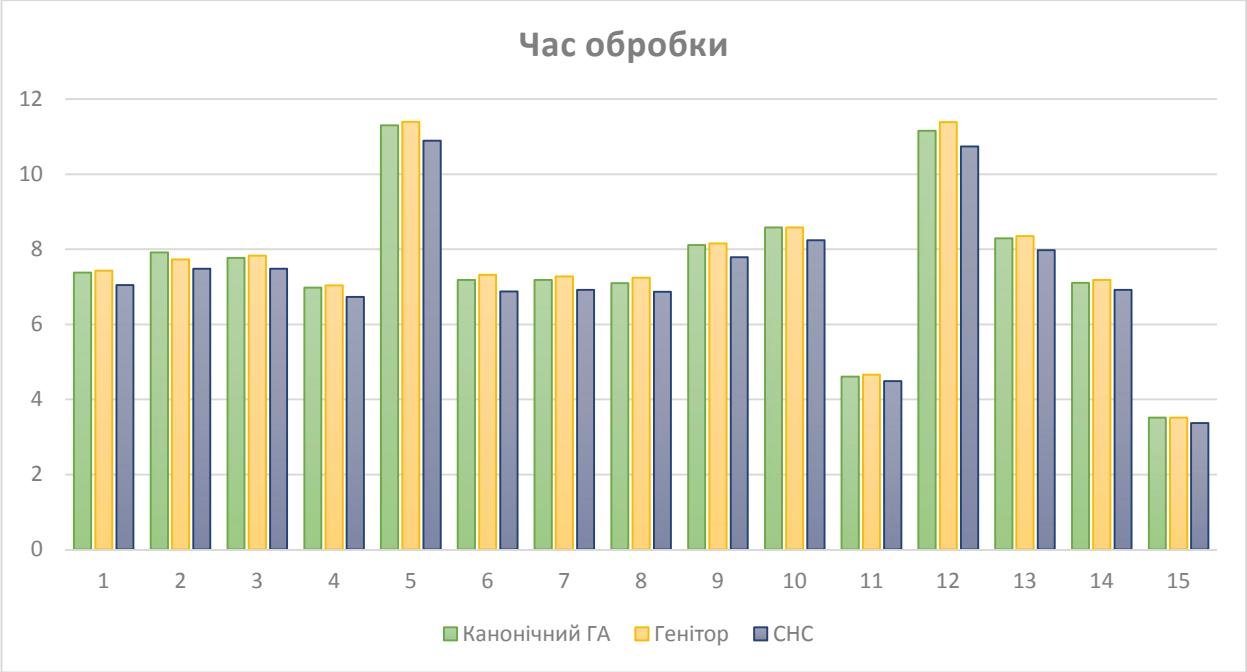


Рисунок 2.4 – час обробки різними моделями ГА

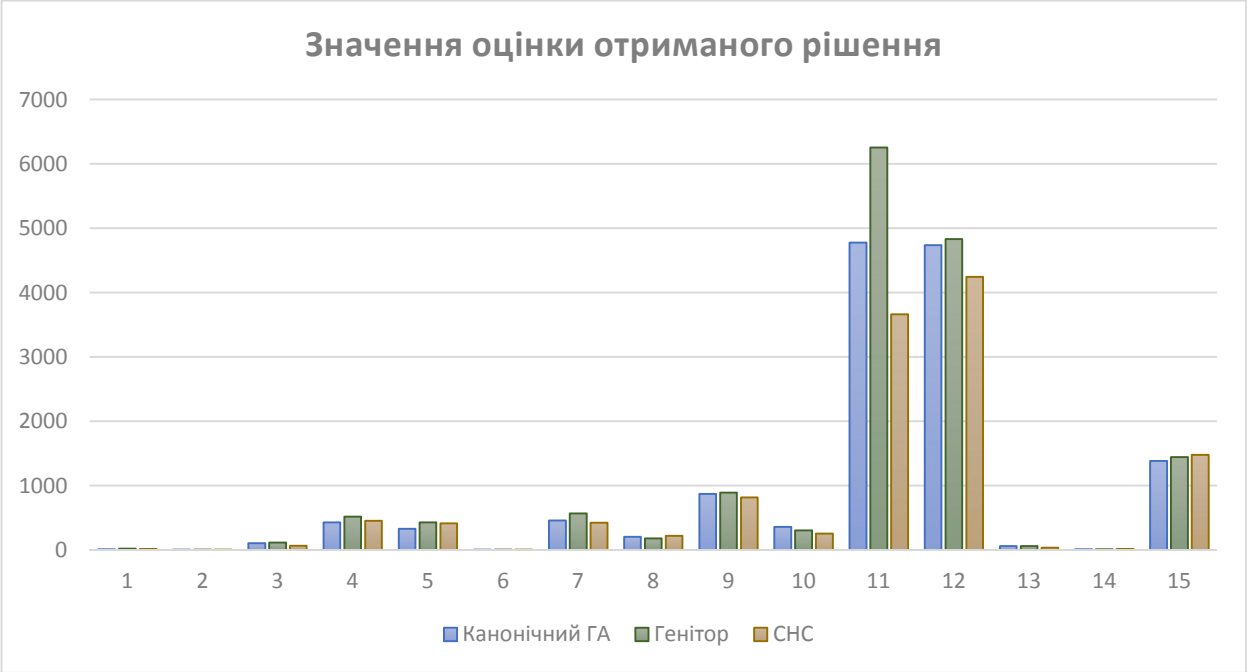


Рисунок 2.5 – значення фітнес-функції отриманого рішення

Таблиця 2.2. Ранги оцінок та моделей

| Номер зображення | Канонічний ГА | | Генітор | | СНС | |
|---|---|---|---|---|---|---|
| | Час | Макс. значення оцінки | Час | Макс. значення оцінки | Час | Макс. значення оцінки |
| 1 | 0,0468 | 0,3592 | 0,0539 | 0,0000 | 0,0000 | 0,2121 |
| 2 | 0,0588 | 0,0363 | 0,0323 | 0,0000 | 0,0000 | 0,3385 |
| 3 | 0,0392 | 0,0643 | 0,0473 | 0,0000 | 0,0000 | 0,4061 |
| 4 | 0,0369 | 0,1719 | 0,0465 | 0,0000 | 0,0000 | 0,1234 |
| 5 | 0,0375 | 0,2318 | 0,0470 | 0,0000 | 0,0000 | 0,0326 |
| 6 | 0,0433 | 0,1587 | 0,0639 | 0,0771 | 0,0000 | 0,0000 |
| 7 | 0,0371 | 0,1953 | 0,0511 | 0,0000 | 0,0000 | 0,2499 |
| 8 | 0,0332 | 0,0679 | 0,0549 | 0,1791 | 0,0000 | 0,0000 |
| 9 | 0,0407 | 0,0233 | 0,0467 | 0,0000 | 0,0000 | 0,0828 |
| 10 | 0,0406 | 0,0000 | 0,0411 | 0,1558 | 0,0000 | 0,2961 |
| 11 | 0,0283 | 0,2358 | 0,0379 | 0,0000 | 0,0000 | 0,4141 |
| 12 | 0,0397 | 0,0196 | 0,0605 | 0,0000 | 0,0000 | 0,1218 |
| 13 | 0,0388 | 0,0000 | 0,0455 | 0,0095 | 0,0000 | 0,3987 |
| 14 | 0,0272 | 0,2142 | 0,0376 | 0,3582 | 0,0000 | 0,0000 |
| 15 | 0,0436 | 0,0628 | 0,0445 | 0,0213 | 0,0000 | 0,0000 |
| Загалом | 0,5917 | 1,8410 | 0,7109 | 0,8010 | 0,0000 | 2,6760 |
| Сума | 2,4327 | | 1,5120 | | 2,6760 | |

Результати, отримані в ході тестування, підтверджують теоретичні припущення, зроблені в таблиці 2.2. Найшвидшим у всіх запусках виявився ГА, реалізований за моделлю СНС. Це пояснюється тим, що витрати часу на оцінку початкової популяції значно знижені в зв’язку з меншим її розміром. Крім того,

знижені витрати часу на роботу операторів ГА, тому що мутація застосовується тільки при перезавантаженні еволюції, а не до кожної з 100 нових осіб.

Алгоритм на основі моделі Генітор виявився найповільнішим, однак різниця в часі обробки дуже мала. Середній час обробки зображення алгоритмом на основі моделі Генітор виявилося на 4,7% більше, ніж у моделі CHC; різниця з канонічним ГА була ще менше і склала менше 1%. Більший час роботи пояснюється великими витратами часу на роботу операторів ГА, тому що з кожної пари батьків виходить тільки один нащадок, а значить селекцію потрібно проводити вдвічі частіше. Також впливає поява тільки одного нового нащадка в кожній популяції: після отримання кожного нащадка потрібно викликати функцію зменшення популяції. Однак витрати часу на роботу операторів генетичного алгоритму значно менші, ніж час перетворення і оцінки зображень, тому й різниця в часі обробки виявилася настільки незначною. Крім того, слід зазначити, що алгоритм Генітор при більшості запусків знаходив найкраще рішення значно раніше, ніж через 100 поколінь, в середньому найкраще рішення було знайдено на 56 поколінні, а значить, вибравши більш відповідний критерій зупинки, можна значно знизити час обробки.

За якістю обробки алгоритм Генітор значно перевершив два інших варіанти реалізації, давши найкращий результат в 9 випадках з 15. Це пояснюється тим, що алгоритм Генітор має найбільш широкий простір пошуку через застосування мутації до кожної одержуваної особи, кросинговеру з високою руйнівною здатністю і відбором нової пари батьків для отримання кожної нової особини.

Алгоритм на основі моделі CHC також використовує оператор кросинговеру з високою руйнівною здатністю, проте відсутність мутації на кожному кроці призводить до значного зниження пристосованості результуючої особи. Дану проблему можна вирішити за допомогою збільшення кількості перезапусків еволюції, проте кожен додатковий перезапуск збільшує час

обробки на 30-50%. Крім того, модель CHC має менший розмір популяції, що спочатку дає менший простір пошуку, ніж у інших моделей.

Сумарний ранг виявився найкращим у моделі Генітор, і вона була обрана для подальших досліджень. Дана модель дає найкращий результат при прийнятному часу обробки. Крім того, як зазначалося вище, час обробки можна скоротити за рахунок зміни критерію зупинки.

## 2.4 Тестування ядра ГА

Як уже було сказано вище, розроблюваний метод вимагає значних обчислювальних ресурсів. Це пояснюється тим, що кожне зображення обробляється поточково, і при отриманні кожного нового нащадка потрібно кілька повних проходів по зображенню для його перетворення і подальшої оцінки. Крім того, значні витрати часу потрібні для початкового обчислення локальних характеристик, використовуваних в ядрі поліпшення зображень, наприклад, середнього значення яскравості кожної точки.

Для визначення найбільш витратних етапів обробки зображення було проведено ряд тестів. Для тестування було обрано 12 зображень розміром 800x800 пікселів. Потім була проведена обробка кожного з цих зображень, і за допомогою програми AMD CodeXL було зареєстровано кількість викликів кожної функції, середній і сумарний час виконання.

Кожне зображення оброблялося алгоритмом на основі моделі Генітор з наступними параметрами:

• Розмір популяції: 20 осіб;

• Кількість поколінь: 100;

• Ймовірність мутації: 30%;

• Оператор кросинговеру: одноточковий;

• Оператор селекції: сигма-відсічення (елітизм).

У таблиці 2.3. наведено час обробки і кількість викликів основних функцій. Час роботи операторів генетичного алгоритму виявився дуже малим, тому в таблиці не наводиться.

Таблиця 2.3

| Етап обробки | Кількість викликів | Середній час (мс) | Сумарний час (мс) | Відсоток від загального часу (%) |
|---|---|---|---|---|
| Отримання каналу яскравості | 12 | 19 | 230 | 0,09 |
| Розрахунок локальних характеристик | 12 | 13297 | 159569 | 63,8 |
| Перетворення зображення | 1440 | 50 | 71683 | 28,66 |
| Оператор Собеля | 1440 | 10 | 14879 | 5,95 |
| Знаходження ентропії | 1440 | 1,6 | 2359 | 0,94 |
| Відновлення зображення | 12 | 115 | 1384 | 0,55 |
| Сума | | | 250104 | |

Як і очікувалося, найбільш обчислювально витратними виявилися функції розрахунку локальних характеристик, перетворення зображення і його оцінки.

Розрахунок середньої яскравості займає 63,8% від загального часу роботи алгоритму. Це пояснюється тим, що крім повного проходу по зображенню необхідно робити прохід по околі кожної точки, а значить обчислювальна складність цього етапу пропорційна добутку кількості точок на зображенні на кількість точок в околі. Розмір околу обирається пропорційно розміру

зображення, тобто обчислювальна складність цього етапу пропорційна квадрату кількості точок на зображенні.

Скорочення часу роботи цього етапу можливе або за рахунок зниження розміру зображення, або за рахунок виключення якихось характеристик з розрахунку.

Другий за складністю етап обробки – перетворення зображення. Виклик цієї функції відбувається один раз для кожної особи, а тому витрати часу на виконання перетворень визначаються кількістю одержуваних осіб і часом виконання одного перетворення.

Нагадаємо, що перетворення зображення відбувається за такою формулою:

$$g(x,y) = T\big(f(x,y)\big) = \left(k\frac{M}{\sigma(x,y)+b}\right)\big(f(x,y) - cm(x,y)\big) + m(x,y)^a; \quad \begin{matrix} x=\overline{0\ldots H_{size}-1}; \\ y=\overline{0\ldots W_{size}-1}; \end{matrix} \tag{15}$$

де $g(x,y)$ і $f(x,y)$ – отримана та початкова яскравість точки з координатами $(x,y)$, $M$ – глобальний середній показник яскравості зображення, $m(x,y)$ – середнє значення яскравості в околі точки з відповідними координатами, $\sigma(x,y)$ – значення середньоквадратичного відхилення в цьому ж околі, $H_{size}$ та $W_{size}$ – висота та ширина зображення відповідно.$a, b, c, k$ – параметри ядра покращення, що визначаються за допомогою генетичного алгоритму.

Глобальне середнє значення освітлення і локальні характеристики обчислюються тільки один раз, а значить, їх використання не дає великого обчислювального навантаження. Найбільш обчислювально складними операціями в даній формулі є піднесення до степеню і обчислення першого множника. Піднесення до степеню необхідно для вирівнювання яскравості зображення, тобто єдиним можливим спрощенням ядра перетворення зображень є спрощення першого складеного множника. Крім того, значного скорочення часу обробки можна домогтися за допомогою скорочення кількості поколінь,

необхідних для знаходження рішення достатньої якості або зменшення розмірів оброблюваного зображення.

Третя функція зі значною обчислювальною складністю – це оператор Собеля. Цей оператор необхідний для оцінки контрастності і чіткості зображення і не може бути виключений з критерію оцінки зображення.

## 2.5 Застосування додаткових методів

Під час тестування було помічено утворення гало та ореолів на границях об’єктів з однорідним освітленням. Щоб пом’якшити ці недоліки було вирішено застосувати вирівнювання гістограми.

Гістограма зображення з $L$ можливими рівнями яскравості у діапазоні $[0, G]$ визначається як дискретна функція

$$h(r_k) = n_k, \tag{16}$$

де $r_k$ – $k$-та інтенсивність у діапазоні $[0, G]$, а $n_k$ – кількість пікселів на зображенні з таким значенням інтенсивності. Проте частіше використовуються нормалізовані гістограми:

$$p(r_k) = \frac{h(r_k)}{n} = \frac{n_k}{n}, \tag{17}$$

де $n$ – загальна кількість пікселів зображення.

Для людського зору більш сприйнятні зображення з великою кількістю різних рівнів яскравості. З цього випливає, що якщо гістограма зображення занадто вузька, то її необхідно розтягнути вздовж горизонтальній осі [34].

Один з методів вирівнювання гістограм був описаний Рафаелем Гонсалесом і його співавторами в [1, 35].

Нехай $p(r_j)$ – нормалізована гістограма вихідного зображення. Тоді для отримання вихідних рівнів яскравості $s_k$ необхідно виконати наступне перетворення:

$$s_k = \sum_{j=1}^{k} p(r_j) = \sum_{j=1}^{k} \frac{n_j}{n}\ , \tag{18}$$

для усіх $k$ = 1, 2, 3, …, $L$, де $s_k$ – значення яскравості $k$-го рівня яскравості на кінцевому зображенні, що відповідає значенню $r_k$ на початковому.

Необхідно зауважити, що при дискретному поданні яскравості в зображенні, $s_k$ буде округлятися в бік найближчого дискретного значення, тобто при низькій кількості точок з деякою яскравістю внаслідок округлення можлива втрата цієї градації.

Крім того, при наявності на зображенні великих рівномірних областей, після обробки сусідні градації яскравості з цих областей будуть значно відрізнятися, що призведе до ступінчастості зображення.

Для виключення таких ефектів було вирішено ввести мінімальне та максимальне обмеження на різницю між сусідніми градаціями яскравості [36]. Для цього вираз було модифіковано таким чином:

$$s_k = \sum_{j=1}^{k} p'(r_j),$$

$$p'(r_j) = \begin{cases} D_{min}, p'(r_j) \le D_{min} \\ p(r_j), D_{min} < p(r_j) \le D_{max}, \\ D_{max}, p(r_j) > D_{max} \end{cases} \tag{19}$$

де $D_{min}$та $D_{max}$ – мінімальне та максимальне обмеження різниці між сусідніми градаціями.

З урахуванням того, що мінімальне обмеження вводиться для виключення можливості втрати інформації, значення $D_{min}$ має дорівнювати $1/L$, де $L$ – кількість можливих рівнів яскравості зображення. Максимальне обмеження обирається дослідником.

Даний метод не змінює кількості різних градацій яскравості, присутніх на зображенні, звідси, ентропія зображення не змінюється, а в ряді випадків може трохи знизитися через втрату градацій з дуже малим значенням гістограми. Проте, за рахунок збільшення різниці між сусідніми градаціями яскравості вдається підвищити кількість і інтенсивність крайових пікселів. Крім того, за рахунок зсуву градацій і розширення гістограми вдається підвищити рівень пристосованості до зору людини за яскравістю.

Витрати часу на обробку за допомогою методу вирівнювання гістограми виявилися вкрай незначними. Обробка одного зображення розміром 800x800 точок в середньому займає 4,9 мс, що є дуже незначним навіть в порівнянні з витратами часу на конвертацію зображення в YCbCr і відновленням зображення.

Таблиця 2.4

| Етап обробки | Кількість викликів | Середній час (мс) | Сумарний час (мс) | Процент від загального часу (%) |
|---|---|---|---|---|
| Конвертація зображення в YCbCr | 12 | 18 | 216 | 13,05 |
| Вирівнювання гістограми | 12 | 4,9 | 59 | 3,56 |
| Відновлення зображення | 12 | 115 | 1380 | 83,38 |
| Усього | | | 1655 | |

## 2.6 Основні результати та висновки до розділу 2

1. Представлена загальна схема застосування генетичного алгоритму для підвищення якості зображень. Розглянуто основні етапи, обраний

алгоритм переходу від кольорового зображення до матриці яскравості і навпаки.

2. Реалізовано і протестовані різні моделі генетичних алгоритмів, оцінена їх ефективність застосування для розв’язання задачі проекту. Обрано алгоритм, на основі моделі Генітор з одноточковим оператором кросинговеру, що дає значно кращу збіжність в контексті досліджуваної задачі, ніж алгоритми, засновані на інших моделях. Краща збіжність моделі Генітор пояснюється широким простором пошуку, що є найбільш важливим в умовах поставленого завдання. Більш низька швидкість роботи може бути компенсована вибором іншого критерію зупинки, оскільки краще рішення знаходиться значно раніше, ніж за допомогою інших алгоритмів. Отримання тільки одного нащадка на кожному кроці робить алгоритм гнучким і процес еволюції можна зупинити на будь-якому етапі.

3. Показано, що найбільш витратними за часом етапами обробки є розрахунок локальних характеристик і перетворення зображення. При цьому розрахунок локальних характеристик виконується тільки 1 раз для кожного зображення і займає в середньому більше 63% загального часу, в той час як перетворення зображення є більш швидким, але викликається для кожної одержуваної особи, що призводить до значних сумарних витрат часу.

# 3. ПРОГРАМНА РЕАЛІЗАЦІЯ, ЇЇ ТЕСТУВАННЯ ТА РЕЗУЛЬТАТИ

У даному розділі будуть наведені результати роботи розробленого програмного продукту, будуть зроблені відповідні висновки.

## 3.1 Аналіз роботи програмного продукту

Експериментальним шляхом було встановлено, що найкращі результати можна отримати при таких параметрах ядра ГА:

- ймовірність ОК – 70%;
- ймовірність мутації – 30%;
- число поколінь – 70;
- розмір популяції – 20 осіб.

Дана конфігурація має суттєвий недолік – великий час виконання, проте, це у повній мірі компенсується отриманим результатом. Розглянемо на прикладі рисунку 3.1 та рисунку 3.2 (наступна сторінка). Рисунок 3.1 – оригінальне зображення, основний недолік – недостатня різкість та контраст. Рисунок 3.2 – зображення, оброблене розробленим алгоритмом. Було усунено загальну розмитість, покращено контраст, проявлено тіні та зроблено більш чіткими текстури об’єктів, зокрема це можна помітити на дверях зліва, мармурових колонах та підлозі.

Мінімальна конфігурація для задовільних результатів:

- ймовірність ОК – 50%;
- ймовірність мутації – 15%;
- число поколінь – 10;
- розмір популяції – 10 осіб.

Рисунок 3.1 – оригінальне зображення [37]

Рисунок 3.2 – оброблене зображення

Не рекомендується задавати параметри менші за вказані, це, як правило, призводить до розмиття зображення, створення ореолів та гало. Мінімальна конфігурація найкраще працює на зображеннях невеликого розміру. Це обумовлено специфікою методу, а саме розрахунками локальних параметрів, оскільки на меншому зображенні вони не призводять до кардинальних змін, тому що у більшості випадків близькі до середніх значень по зображенню. Приклад роботи такої конфігурації наведено на рисунку 3.3.

Рисунок 3.3 – Оригінальне зображення (зліва) [37], оброблене зображення (справа)

## 3.2 Основні результати та висновки до розділу 3

1. Було наведено оптимальну та мінімальну конфігурацію для ядра ГА, проаналізовано їх роботу.
2. Було наведено результати роботи програмного продукту на оптимальній та на мінімальній конфігураціях.

# 4. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для покращення якості зображень, використовуючи еволюційні обчислення. Програмний продукт був розроблений за допомогою мови програмування C++ у середовищі розробки Visual Studio Community 2015. Інтерфейс користувача створений за допомогою технології WindowsForms.

Програмний продукт є крос-платформним та рекомендується для використання на персональних комп'ютерах під управлінням операційних систем Windows та Linux.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

## 4.1 Постановка задачі

У роботі застосовується метод ФВА. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для вибору методу прогнозування місцезнаходження об’єктів у контекстно-залежних системах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;

– забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

### 4.1.1 Обґрунтування функцій програмного продукту

Головна функція $F_0$– розробка програмного продукту, який будує набір моделей прогнозу та перевіряє їх точність на певному наборі даних. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

$F_1$ – вибір мови програмування;

$F_2$ – використання готових бібліотек;

$F_3$ – вибір фреймворку.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція $F_1$:

а) мова програмування C++;

б) мова програмування Java;

Функція $F_2$:

а) написання алгоритмів вручну;

б) використання готових бібліотек;

Функція $F_3$:

а) Qt;

б) Miscrosoft Windows Studio Community 2015.

### 4.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 4.1).

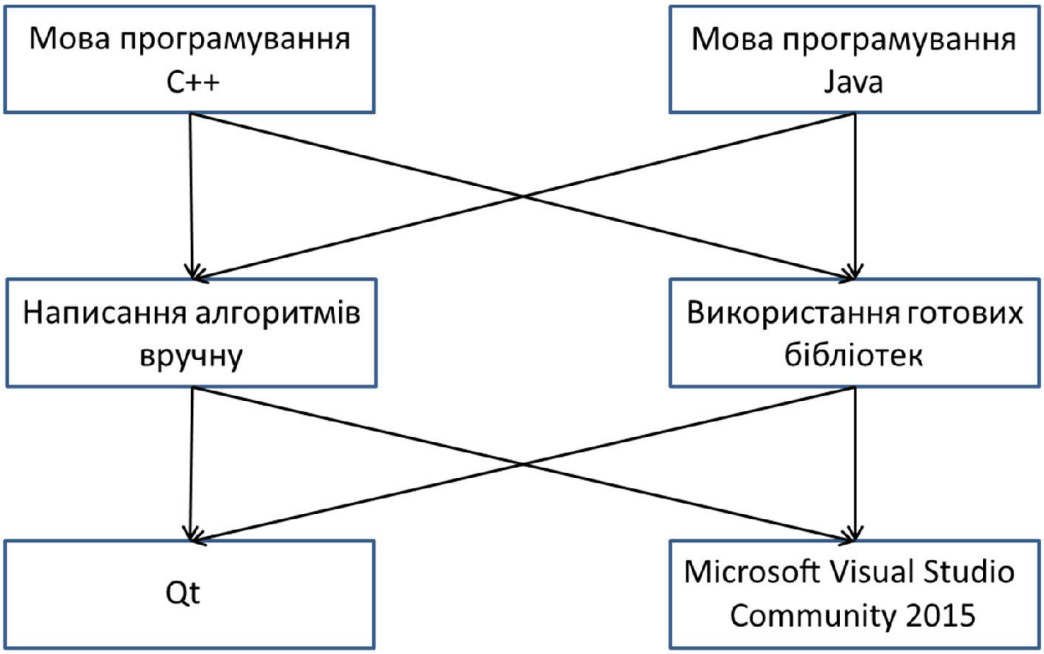


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 4.1 – Позитивно-негативна матриця

| **Основні функції** | **Варіанти реалізації** | **Переваги** | **Недоліки** |
|---|---|---|---|
| *F1* | *А* | Займає менше часу при написанні коду | Код повільніше виконується |
| | *Б* | Код швидше виконується | Займає більше часу при написанні коду |
| *F2* | А | Більша гнучність у використанні | Займає більше часу при написанні коду |
| | Б | Займає менше часу при написанні коду | Менша гнучкість у використанні |
| *F3* | *А* | Швидший на етапі створення | Потребує стороннє ПО для аналізу |
| | *Б* | Має багато опцій для дослідження роботи ПО | Витрачає більше апаратних ресурсів |

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам.

Функція *F1:*

Оскільки проводиться оцінка великої кількості методів, потрібно швидко їх реалізовувати, тому варіант б) має бути відкинутий.

Функція *F2:*

Оскільки методи написані вручну та за допомогою готових бібліотек будуть давати однакові результати, вважаємо варіанти а) та б) гідними розгляду.

Функція *F3:*

Точна оцінка технічних параметрів роботи створеного ПО важливіша для даної роботи, ніж зручність створення, тому варіант а) має бути відкинутий.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3б
2. F1а – F2б – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

## 4.2 Обґрунтування системи параметрів ПП

### 4.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – швидкодія мови програмування;

– *X2* – об’єм пам’яті для збереження даних;

– *X3* – час обробки даних;

– *X4* – потенційний об’єм програмного коду.

*X1*: Відображає швидкодію операцій залежно від обраної мови програмування.

*X2:* Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

*X3:* Відображає час, який витрачається на дії.

*X4:* Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

### 4.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

Таблиця 4.2 – Основні параметри ПП

| **Назва Параметра** | **Умовні позначення** | **Одиниці виміру** | **Значення параметра** | | |
|---|---|---|---|---|---|
| | | | **гірші** | **середні** | **кращі** |
| Швидкодія мови програмування | X1 | Оп/мс | 2000 | 11000 | 19000 |
| Об’єм пам’яті для збереження даних | X2 | Мб | 32 | 16 | 8 |
| Час обробки даних алгоритмом | X3 | мс | 800 | 420 | 60 |
| Потенційний об’єм програмного коду | X4 | кількість строк коду | 2000 | 1500 | 1000 |

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

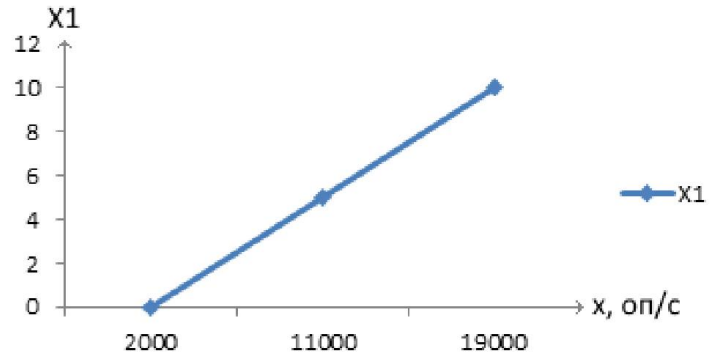


Рисунок 4.2 – X1, швидкодія мови програмування

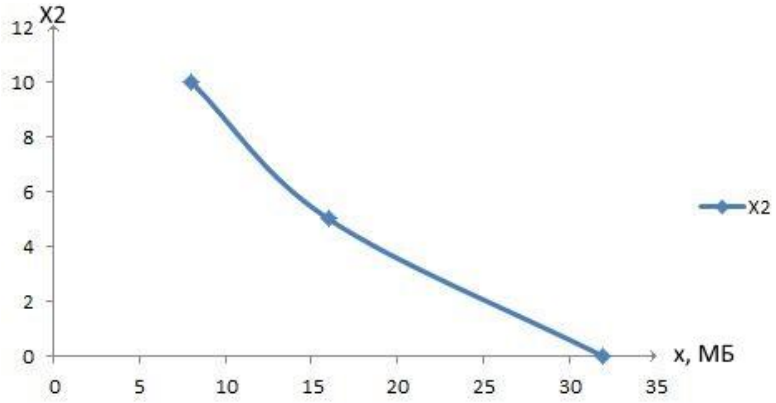


Рисунок 4.3 – X2, об’єм пам’яті для збереження даних

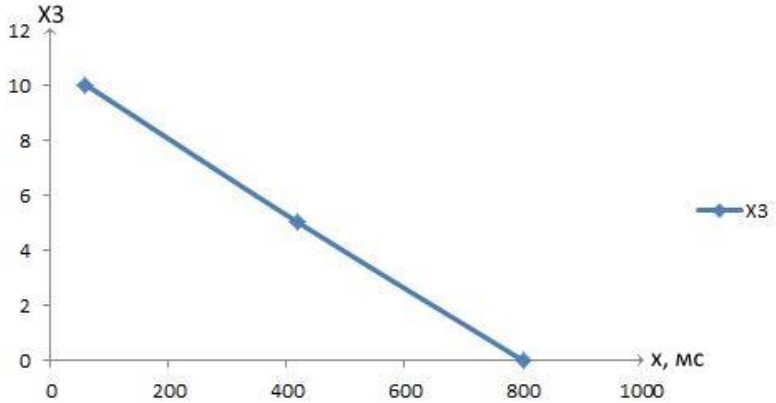


Рисунок 4.4 – X3, час обробки даних алгоритмом

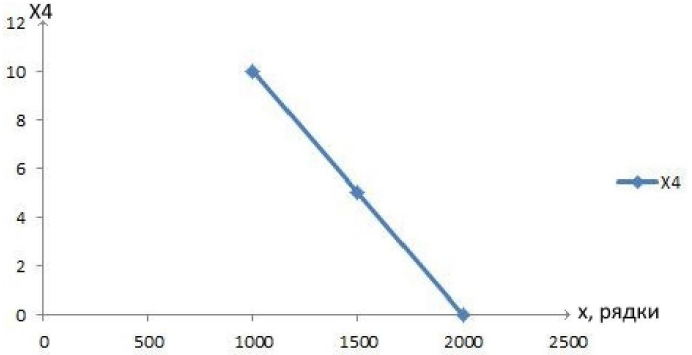


Рисунок 4.5 – X4, потенційний об’єм програмного коду

### 4.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

| Позначення параметра | Назва параметра | Одиниці виміру | Ранг параметра за оцінкою експерта | | | | | | | Сума рангів $R_i$ | Відхилення $\Delta_i$ | $\Delta_i^2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | |
| *X1* | Швидкодія мови програмування | Оп/мс | 4 | 3 | 4 | 4 | 4 | 4 | 4 | 27 | 0.75 | 0.56 |
| *X2* | Об’єм пам’яті для збереження даних | Мб | 4 | 4 | 4 | 3 | 4 | 3 | 3 | 25 | -1.25 | 1.56 |
| *X3* | Час обробки даних алгоритмом | Мс | 2 | 2 | 1 | 2 | 1 | 2 | 2 | 12 | -14.25 | 203.06 |
| *X4* | Потенційний об’єм програмного коду | кількість строк коду | 5 | 6 | 6 | 6 | 6 | 6 | 6 | 41 | 14.75 | 217.56 |
| | Разом | | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 105 | 0 | 420.75 |

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105,$$

де $N$ – число експертів, $n$ – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26.25$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420.75$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420.75}{7^2(5^3 - 5)} = 1{,}03 > W_k = 0.67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0.67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця4.4 – Попарне порівняння параметрів

| Параметри | Експерти | | | | | | | Кінцева оцінка | Числове значення |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | |
| X1 і X2 | = | > | = | < | = | < | < | < | 0.5 |
| X1 і X3 | < | < | < | < | < | < | < | < | 0.5 |
| X1 і X4 | > | > | > | > | > | > | > | > | 1.5 |
| X2 і X3 | < | < | < | < | < | < | < | < | 0.5 |
| X2 і X4 | > | > | > | > | > | > | > | > | 1.5 |
| X3 і X4 | > | > | > | > | > | > | > | > | 1.5 |

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, $a_{ij}$ визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{де } b_i = \sum_{i=1}^{N} a_{ij}.$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j.$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

| Параметри $x_i$ | Параметри $x_j$ | | | | Перша ітер. | | Друга ітер. | | Третя ітер | |
|---|---|---|---|---|---|---|---|---|---|---|
| | X1 | X2 | X3 | X4 | $b_i$ | $K_{\text{ві}}$ | $b_i^1$ | $K_{\text{ві}}^1$ | $b_i^2$ | $K_{\text{ві}}^2$ |
| X1 | 1.0 | 0.5 | 0.5 | 1.5 | 3.5 | 0.219 | 22.25 | 0.216 | 100 | 0.215 |
| X2 | 1.5 | 1.0 | 0.5 | 1.5 | 4.5 | 0.281 | 27.25 | 0.282 | 124.25 | 0.283 |
| X3 | 1.5 | 1.5 | 1.0 | 1.5 | 5.5 | 0.344 | 34.25 | 0.347 | 156 | 0.348 |
| X4 | 0.5 | 0.5 | 0.5 | 1.0 | 2.5 | 0.156 | 14.25 | 0.155 | 64.75 | 0.154 |
| Всього: | | | | | 16 | 1 | 98 | 1 | 445 | 1 |

## 4.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об’єм пам’яті для збереження даних) *X1* (швидкодія мови програмування) та *X3* відповідають технічним вимогам умов функціонування даного ПП.1

Абсолютне значення параметра *X4 (*потенційний об’єм програмного коду) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 1800 або варіанту б) 1200.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} ,$$

де $n$ – кількість параметрів;$K_{вi}$– коефіцієнт вагомості $i$–го параметра;$B_i$ – оцінка $i$–го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

| Основні функції | Варіант реалізації функції | Абсолютне значення параметра | Бальна оцінка параметра | Коефіцієнт вагомості параметра | Коефіцієнт рівня якості |
|---|---|---|---|---|---|
| F1(X1) | А | 11000 | 3.6 | 0.215 | 0.774 |
| F2(X3) | А, Б | 800 | 2.4 | 0.348 | 0.835 |
| F2(X4) | А | 1800 | 2 | 0.154 | 0.308 |
| | Б | 1200 | 8 | 0.154 | 1.232 |
| F3(X2) | Б | 16 | 3.4 | 0.283 | 0.962 |

За даними з таблиці 4.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

$K_{K1} = 0.774 + 0.835 + 0.308 + 0.962 = 2.879$

$K_{K2} = 0.774 + 0.835 + 1.232 + 0.962 = 3.803$

Як видно з розрахунків, кращим є другий варіант, для якого коефіцієнт технічного рівня має найбільше значення.

## 4.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

При цьому варіант 3 має додаткове завдання:

3. Реалізація методів аналізу;

А варіант 4 має інше додаткове завдання:

4. Обробка інтерфейсу готових бібліотек.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 2.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.1)$$

де $T_P$ – трудомісткість розробки ПП;$K_П$ – поправочний коефіцієнт;$K_{СК}$ – коефіцієнт на складність вхідної інформації; $K_M$ – коефіцієнт рівня мови програмування;$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_P$ =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм другої групи складності, степінь новизни Б), тобто $T_P$ =27 людино-днів, $K_П$ =0.9,$K_{СК}$ = 1,$K_{СТ}$ =0.8:

$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44$ людино-днів.

Для третього завдання (використовується алгоритм другої групи складності, ступінь новизни Г з використанням перемінної інформації):

Тр = 12 людино-днів;

Кп = 0.72; Кст = 0.8;

То = 12 • 0.72 • 0,8 = 6.91.

Для четвертого завдання (використовується алгоритм третьої групи складності, ступінь новизни Г):

Тр = 8 людино-днів;

Кп = 0.6; Кст = 1;

То = 8 • 0.6 • 1 = 4.8.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$Т_I$ = (122.4 + 19.44 + 6.91) · 8 = 1190 людино-годин;

$Т_{II}$ = (122.4 + 19.44 + 4.8) · 8 = 1173.12 людино-годин;

Найбільш високу трудомісткість має варіант І.

В розробці беруть участь два програмісти з окладом 6000 грн., один фінансовий аналітик з окладом 9000 грн. Визначимо зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} грн.,$$

де М – місячний оклад працівників; $T_m$ – кількість робочих днів тиждень; $t$ – кількість робочих годин в день.

$$С_ч = \frac{6000 + 6000 + 9000}{3 \cdot 21 \cdot 8} = 41.67 грн.$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д,$$

де $С_ч$– величина погодинної оплати праці програміста; $Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 41.67 \cdot 1190 \cdot 1.2 = 59504.76$ грн.

II. $С_{ЗП} = 41.67 \cdot 1173.12 \cdot 1.2 = 58660.69$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 59504.76 \cdot 0.22 = 13091.05$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 58660.69 \cdot 0.22 = 12905.35$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 6000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 6000 \cdot 0.2 = 14400$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 14400 \cdot (1 + 0.2) = 17280$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22 = 17280 \cdot 0.22 = 3801.6$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 8000 = 2300$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 8000 \cdot 0.05 = 460$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706.4 \cdot 0.156 \cdot 0.2 \cdot 2.0218 = 107.64 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 8000 \cdot 0.67 = 5360 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$$С_{ЕКС} = 17280 + 3801.6 + 2300 + 460 + 107.64 + 5360 = 29309.24 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 29309.24 / 1706.4 = 17.18 \text{ грн/час.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М\text{-}Г} \cdot Т$$

I. $С_М = 17.18 \cdot 1190 = 20444.2$ грн.;

II. $С_М = 17.18 \cdot 1173.12 = 20154.2$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0.67$$

I. $С_Н = 59504.76 \cdot 0.67 = 39868.19$ грн.;

II. $С_Н = 58660.69 \cdot 0.67 = 39302.66$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

I. $С_{ПП}$ = 59504.76 + 13091.05 + 20444.2+ 39868.19 = 132908.2 грн.;

II. $С_{ПП}$ = 58660.69 + 12905.35 + 20154.2 + 39302.66 = 131022.9 грн.;

## 4.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj},$$

$К_{ТЕР1}$ = 2.879 / 132908.2 = $0.22 \cdot 10^{-4}$;

$К_{ТЕР2}$ = 3.803 / 131022.9= $0.29 \cdot 10^{-4}$;

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0.29 \cdot 10^{-4}$.

## 4.6 Висновки до розділу 4

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що

залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0.29 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – C++;

– використання готових бібліотек;

– фреймворк Microsoft Visual Studio Community 2015.

За умови подальшого розвитку паралельних обчислень та удосконаливши критерій оцінки отриманого зображення, даний продукт дійсно матиме економічну цінність. Його можливо використовувати як елемент комп'ютерного зору для попередньої обробки зображень. Основна перевага розробки – невибагливість до вхідних даних, чого не можна сказати про детерміновані та евристичні методи.

Незважаючи на це, ефективність програмного продукту занадто низька для успішного виходу на ринок – необхідно обробляти відеоряд у режимі, наближеному до реального часу, при якості відео хоча би в 1 мегапіксель (кадр розміром 1366*768 пікселів), розроблений продукт обробляє таке зображення приблизно 2 секунди.

Зваживши усі ці аспекти, можна сказати, що програмний продукт потребує суттєвого доопрацювання, щоб бути конкурентоспроможним.

# ВИСНОВКИ

Дипломна робота присвячена дослідженню методів еволюційних обчислень для обробки зображень та розробки програмних засобів для автоматичного покращення візуальної якості цифрових зображень, що базуються на генетичному алгоритмі.

У результаті виконання дипломної роботи були отримані такі результати:

1. Розроблена модифікована комплексна оцінка якості зображень. Дослідження показали, що дана оцінка враховує як основні критерії якості зображень (контраст, повнота використання можливих градацій яскравості), так і особливості сприйняття візуальної інформації людиною (рівень адаптації до зору людини за яскравістю). Крім того, враховуються деякі особливості функціонування алгоритму.

2. Запропоновано використовувати в якості основи для проектування ГА модель Генітор з одноточковим оператором кросинговеру. У результаті експериментів встановлено, що дана модель найбільш точно задовольняє потреби поставленої задачі в силу найбільш широкого кола пошуку.

3. Запропоновано використовувати метод вирівнювання гістограми замість середньоквадратичного відхилення. Це дозволило скоротити час виконання приблизно на 30% й одночасно покращило результат обробки.

4. Спроектовано й реалізовано програмний засіб обробки зображень з використанням розробленого ГА.

5. Виконано економічний аналіз розробки.

Для подальшого розвитку необхідно удосконалити критерій оцінки, аби виключити утворення гало навколо об’єктів з рівномірним освітленням. Також необхідно оптимізувати алгоритм перетворення Y-каналу, аби обробка одного зображення в 1,5 мегапікселя займала не більше 0,5 с. Тоді програмний засіб можливо буде використовувати в якості елементу комп’ютерного зору. Можна

зробити припущення, що даний алгоритм краще комбінувати з детермінованими методами, наприклад, сегментувавши зображення, застосувати алгоритм до його частин враховуючи для рівня освітлення в околі лише пікселі виділеного об’єкту. Таким чином можна буде вирішити проблему гало та ореолів, тому що для обчислень не будуть використані пікселі, які об’єкту не належать (крім рівня загального освітлення).

Основна потенційна галузь застосування – аерофотозйомка. Використовуючи розробку можливо збільшити відстань до земної поверхні без погіршення якості, таким чином збільшивши ефективний розмір носія інформації.

# ПЕРЕЛІК ПОСИЛАНЬ


1. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. - М.: Техносфера, 2005. - 1072 с.
2. Батищев, Д.И. Оптимизация многоэкстремальных функций с помощью генетических алгоритмов/ Батищев Д.И., Исаев С.А. - ВГТУ, 1997.
3. Цой, Ю.Р. Трехэтапная обработка цифровых изображений с использованием эволюционирующих искусственных нейронных сетей / Ю.Р. Цой, В.Г. Спицын // Всероссийская научная конференция по нечетким системам и мягким вычислениям НСМВ-2006 (20-22 сентября 2006г., Тверь): Труды конференции. - М.: Физматлит, 2006. - С. 231-244.
4. Курейчик, В.М. Эволюционные вычисления: генетическое и эволюционное программирование / В.М. Курейчик, С.И. Родзин // Новости искусственного интеллекта. - 2003. - № 5(59).
5. Hinterding, R. Gaussian mutation and self-adaptation in numeric genetic algorithms/ R. Hinterding // IEEE International Conference on Evolutionary Computation. - IEEE Press, 1995. - P. 384-389.
6. Белоусов, А.А. Применение генетических алгоритмов и вейвлет-преобразований для повышения качества изображений/ А.А. Белоусов, В.Г. Спицын, Д.В. Сидоров // Известия Томского политехнического университета, Т. 309. № 7. - 2006. - С. 21-26.
7. Каширина, И.Л. Введение в эволюционное моделирование. Учебное пособие/ И.Л. Каширина. Воронеж: ВГУ, 2007.
8. Емельянов, В.В. Теория и практика эволюционного моделирования/ В.В. Емельянов, В.В. Курейчик, В. М. Курейчик. — М.: ФИЗМАТЛИТ, 2003. - 432 с.

9. Курейчик, В.М. Параллельный генетический алгоритм. Модели и проблемы построения/ В.М. Курейчик, Кныш Д.С. - Таганрог: Технологический ин-т ЮФУ, 2010.

10. Курейчик В. М. Генетические алгоритмы и их применение/ В.М. Курейчик. - Таганрог: Изд-во ТРТУ, 2002.

11. Baeck, T. Evolutionary computation/ T. Baeck, D. Fogel, Z. Michalewicz. - Berlin Heidelberg: Springer-Verlag, 2000.

12. De Jong, K.A. An analysis of the behavior of a class of genetic adaptive systems: Unpublished PhD thesis / K. De Jong. - University of Michigan, Ann Arbor, 1975. - Also University microfilms No. 76-9381 – Режим доступу: http://www.cs.gmu.edu/~eclab Дата доступу – 30.04.2016.

13. Syswerda, G. Uniform crossover in genetic algorithms/ G. Syswerda // Proceedings of Third International Conference on Genetic Algorithms. - San Mateo, CA: Morgan Kaufmann, 1989. - P.2-9.

14. Holland, J.H. Adaptation in natural and artificial systems/ J.H. Holland. - Michigan: The University of Michigan Press, 1975.

15. Журавель И.М. Краткий курс теории обработки изображений Режим доступу: http://matlab.exponenta.ru/imageprocess/book2/index.php Дата доступу – 5.05.2016.

16. Whitley, D. An Overview of Evolutionary Algorithms: Practical Issues and Common Pitfalls/ D. Whitley // Journal of Information and Software Technology. - 2001.

17. Whitley, D. Genetic Algorithms and Neural Networks: Optimizing Connections and Connectivity/ D. Whitley, T. Starkweather, C. Bogart // Parallel Computing, 1990, Vol. 14, pp. 341-361. – Режим доступу: http://www.cs.colostate.edu/~genitor/ Дата доступу – 7.05.2016.

18. Whitley, D. Genitor: A different genetic algorithm/ D. Whitley, J. Kauth // Proceedings of the Rocky Mountain Conference on Artificial Intelligence. - Denver, CO, 1988. - P. 118-130.

19. Eshelman, L.J. The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination/ L.J. Eshelman, G.J.E. Rawlins // Proceedings of the First Workshop on Foundations of Genetic Algorithms. - Morgan Kaufmann, 1991. - P. 265-283.

20. Спицын, В.Г. Улучшение качества изображений на основе применения эволюционирующей нейронной сети, вейвлет-преобразования и генетического алгоритма/ В.Г. Спицын, Ю.Р. Цой, А.В. Чернявский, А.А. Белоусов, Д.В. Сидоров // Труды Российского научно-технического общества радиотехники, электроники и связи имени А.С. Попова. Серия: Цифровая обработка сигналов и ее применение. Выпуск: IX-2, Доклады 9-й Международной конференции "Цифровая обработка сигналов и ее применение", 28-30 марта 2007, Москва, 2007. С. 570-574.

21. Belousov, A.A. Application of Wavelet Transform and Genetic Algorithms for Image Processing/ A.A. Belousov, V.G. Spitsyn, D.V. Sidorov // Proceedings of International Conference on Image Processing, Computer Vision, &Pattem Recognition, Las Vegas, USA, July 13-16, 2009, Vol. 2.-P. 846-851.

22. Belousov, A.A. Applying wavelets and evolutionary algorithms to automatic image enhancement/A.A. Belousov, D.V. Sidorov, V.G. Spitsyn // XIII International Symposium "Atmospheric and Ocean Optics. Atmospheric Physics", Tomsk, July 2-6, 2006. - P. 104.

23. Munteanu, C. Gray-Scale Image Enhancement as an Automatic Process Driven by Evolution/ C. Munteanu, A. Rosa // IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics. - Vol. 34. - 2004.-P. 1292-1298.

24. Панченко Т.В. Генетические алгоритмы: учебно-методическое пособие/ Т.В. Панченко; под ред. Ю.Ю. Тарасевича. - Астрахань: Издательский дом «Астраханский университет», 2007. - 87 с.

25. Белоусов, А.А. Двухэтапный метод улучшения изображений/ А.А. Белоусов // Технологии Microsoft в теории и практике программирования: Сборник трудов VI Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых - Томск, 17-18 марта 2009- Томск: ТПУ. - 2009. - с. 123-125.

26. Белоусов, А.А. Применение генетических алгоритмов для повышения качества полутоновых изображений/ А.А. Белоусов, В.Г. Спицын // Молодежь и современные информационные технологии: Сборник трудов IV Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых - Томск, ТПУ, 28 февраля - 2 марта 2006. - Томск: ТПУ. - 2006. - с. 411-413.

27. Линдли, К. Практическая обработка изображений на языке Си/ К. Линдли ; [пер. с англ. А.А. Брюзгина] - М.: Мир, 1996.

28. Цой, Ю.Р. Нейроэволюционный алгоритм и программные средства для обработки изображений. Диссертация на соискание ученой степени кандидата наук/ Ю.Р. Цой. - Томск, 2007

29. Baeck, T. An overview of parameter control methods by self- adaptation in evolutionary algorithms/ T. Baeck // Fund. Inform. - 1998. -Vol. 35, no. 1-4.-P. 51-66.

30. Baeck, T. Self-adaptation/ T. Baeck // Handboook of Evolutionary Computation. - Bristol: Institute of Physics Publishing; New York: Oxford University Press, 1997. - C7.1:1-15.

31. Baeck, T. Self-adaptation in genetic algorithms/ T. Baeck // Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on

Artificial Life / eds. F.J. Varela, P. Bourgine. - Cambridge, MA: MIT Press, 1992. - P.263-271.

32. Baker, J.E. Adaptive selection methods for genetic algorithms/ J.E. Baker // Proceedings of the International Conference on Genetic Algorithms and Their Applications, 1985. - P. 101-111.

33. Eiben, A.E. Evolutionary algorithms with on-the-fly population size adjustment/ A.E. Eiben, E. Marchiori, V.A. Valko // Parallel Problem Solving from Nature, PPSN VIII, LNCS Vol. 3242 / eds. X. Yao [et al.]. - Berlin: Springer-Verlag, 2004. - P. 41-50.

34. Белоусов, А.А. Высокоскоротной метод повышения качества изображений/ А.А. Белоусов // Современные техника и технологии: Труды XIV Международной научно-практической конференции студентов, аспирантов и молодых ученых - Томск, 24-28 марта 2008. - Томск: ТПУ. – с. 244-245.

35. Gonzalez, R.C. Digital Image Processing Using MATLAB/ R.C. Gonzalez, R.E. Woods, S.L. Eddins. - Prentice Hall, 2004

36. Белоусов, А.А. Двухэтапный метод улучшения изображений/ А.А. Белоусов, В.Г. Спицын // Труды XIII Международной научно-практической конференции студентов, аспирантов и молодых ученых "Современная техника и технологии ". Т. 2. Томск. 26 - 30 марта 2007 г. - Изд-во ТПУ. - 2007. - С. 282-284.

37. Сайт бібліотеки обробки зображень CImg, стандартний пакет зображень для тестів (у складі бібліотеки). – Режим доступу: http://cimg.eu/download.shtml – Дата доступу: 03.03.2016.