Вінницький національний технічний університет
Факультет комп'ютерних систем та автоматики
Кафедра автоматизації та інтелектуальних інформаційних технологій

# Пояснювальна записка

до бакалаврської дипломної роботи

бакалавр
(освітньо-кваліфікаційний рівень)

на тему: Дослідження роботи острівного генетичного алгоритму

Виконав: студент 4 курсу, групи 1СІ-16б
Спеціальність 151 – Автоматизація та комп'ютерно-інтегровані технології
Опанасюк В. Є.

Керівник: к.т.н., доцент каф. АІІТ
Іванов Ю. Ю.

Рецензент: _____

Вінниця ВНТУ – 2020 року

Вінницький національний технічний університет

Факультет комп'ютерних систем і автоматики

Кафедра автоматизації та інтелектуальних інформаційних технологій

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 151 – «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ
Завідувач кафедрою АІІТ
д.т.н., проф. Квєтний Р. Н.
«___»_____ 202_ р.

**ЗАВДАННЯ**

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Опанасюку Владиславу Євгеновичу

1. Тема роботи: Дослідження роботи острівного генетичного алгоритму.
   Керівник роботи: к.т.н., доцент каф. АІІТ Іванов Ю.Ю.
   Затверджені наказом ВНТУ від «30» 08 2019 року № 1.
2. Строк подання роботи студентом: «20» 06 2020 р.
3. Вихідні дані до роботи: набір тестових функцій; точність обчислень; пакет *Parallel Computing Toolbox* середовища *MatLab*; кількість ядер процесора – не менше 4; острівна модель генетичного алгоритму (задано кількість поколінь; розмір популяції; довжину хромосом; тип кроссоверу та мутації; принцип формування покоління).
4. Зміст розрахунково-пояснювальної записки: аналітичний огляд еволюційних алгоритмів; розробка математичної моделі острівного генетичного алгоритму; розробка програмного забезпечення та експериментальні дослідження.
5. Перелік графічного матеріалу: структура еволюційної стратегії; схема програми; інтерфейс програмного забезпечення.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1-3 | Іванов Ю. Ю.,<br>к.т.н., доцент каф. АІІТ | | |

7. Дата видачі завдання: «10»___09___ 2019 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналітичний огляд еволюційних алгоритмів | до 10.11.2019 | |
| 2 | Розробка математичної моделі острівного генетичного алгоритму | до 10.02.2020 | |
| 3 | Розробка програмного забезпечення та експериментальні дослідження | до 10.05.2020 | |
| 4 | Оформлення пояснювальної записки і графічного матеріалу | до 30.05.2020 | |
| 5 | Попередній захист роботи | до 04.06.2020 | |
| 6 | Остаточний захист роботи | до 20.06.2020 | |

**Студент** _____ (підпис) Опанасюк В. Є. (прізвище та ініціали)

**Керівник роботи** _____ (підпис) Іванов Ю. Ю. (прізвище та ініціали)

# АНОТАЦІЯ

У даній роботі досліджується острівна модель генетичного алгоритму, яка використовується для пошуку екстремуму нелінійної функції. Проаналізовано математичну основу та показано основні особливості роботи з нею. Розроблено алгоритмічне та програмне забезпечення, яке дозволяє розв'язати поставлену задачу та виконати експериментальні дослідження.

# ABSTRACT

In this work has been researched an island model of genetic algorithm, which is used to find the extremum of the nonlinear function. The mathematical basis has been analyzed and the main features of its work have been shown. The algorithms and software, that allows to solve a given task and to perform experimental researches, have been developed.

# ЗМІСТ

# ВСТУП

*Актуальність.* У наш час багато теорій розглядається крізь призму штучного інтелекту. З них виділяється еволюційне моделювання – напрямок дослідження штучного інтелекту, в основу якого покладено запозичені з популяційної генетики принципи, які дозволяють виконувати моделювання природних еволюційних процесів на комп'ютері для розв'язання різноманітних задач. Дані методи застосовують у випадку, коли задачу неможливо розв'язати точними математичними методами через неповноту вхідних та вихідних даних, наявність обмежень, високої розмірності, зовнішніх умов, які динамічно змінюються та функціонують у конкурентному середовищі тощо [1-5].

У даній роботі розглядається генетичний алгоритм, який застосовано для оптимізації нелінійної функції від багатьох змінних. Механізми, які закладені в основу даного алгоритму, не можуть гарантувати знаходження найкращого рішення, проте, вони з великою ймовірністю знайдуть субоптимальне (майже оптимальне) рішення. Тому їх часто застосовують у поєднанні з класичними методами оптимізації, оскільки така комбінація, при вдалому налаштуванні параметрів алгоритму, дозволяє досягти високої ефективності розв'язання заданої задачі. Але проблема ефективного аналізу простору пошуку та отримання необхідної точності залишається актуальною [6-8].

Останнім часом, у зв'язку зі швидким розвитком обчислювальної техніки, популярною стає концепція паралельних обчислень, яка дозволяє виконувати декілька дій одночасно, що є перспективним для використання у еволюційних алгоритмах. Великі задачі можна розділити на декілька менших, а потім кожну з них розв'язати незалежно від інших. За цим напрямком спостерігається збільшення кількості наукових праць, зокрема В.В. Курейчика [9-12], І.М. Кравця [13], М. Мікі [28], С.Н. Котряхової [30] та інших. Вони акцентують увагу на острівній моделі генетичних алгоритмів, вважаючи її ефективною та зручною для пошуку глобального екстремуму функції; оптимізації динамічних систем; налаштування нейромереж; пошуку найменших маршрутів тощо.

У зв'язку з цим актуальною є науково-практична задача дослідження роботи острівного генетичного алгоритму у ході розв'язання певної задачі, наприклад, оптимізації функції.

*Мета і задачі дослідження.* Метою роботи є підвищення ефективності розв'язання задачі оптимізації нелінійних функцій від багатьох змінних за рахунок використання острівної моделі генетичного алгоритму.

Для досягнення мети необхідно розв'язати наступні задачі:

– проаналізувати основи теорії еволюційних алгоритмів;

– дослідити математичну модель острівного генетичного алгоритму;

– розробити програмні засоби та оцінити ефективність роботи алгоритму.

*Об'єктом дослідження* є оптимізація функції багатьох змінних.

*Предметом дослідження* є моделі, методи та інструментальні засоби для розв'язання задачі пошуку екстремуму функції багатьох змінних.

*Методи дослідження.* У роботі використано поняття теорії оптимізації та еволюційних обчислень, а також концепцію паралелізму під час дослідження роботи генетичних алгоритмів у ході розв'язання задачі пошуку екстремуму функції багатьох змінних. Для аналізу та перевірки достовірності отриманих теоретичних положень застосовано експериментальне дослідження.

*Науково-технічний результат роботи* полягає у тому, що на основі теоретичних положень розроблене математичне, алгоритмічне та програмне забезпечення, яке дозволяє оптимізувати функцію багатьох змінних з використанням генетичних алгоритмів.

У роботі отримані результати, які мають *практичну цінність*, а саме: наведено експериментальні дослідження ефективності острівного генетичного алгоритму для розв'язання задачі оптимізації функції від багатьох змінних.

*Апробація результатів та публікації.* За результатами даної роботи опубліковано 1 тези доповіді [14] на XLIX науково-технічній конференції Факультету комп'ютерних систем і автоматики (м. Вінниця, 2020).

# 1 АНАЛІТИЧНИЙ ОГЛЯД ЕВОЛЮЦІЙНИХ АЛГОРИТМІВ

## 1.1 Вступ до еволюційної теорії

Основний механізм еволюції — це природний відбір. Його суть полягає в тому, що більш пристосовані особини мають більше можливостей для виживання і розмноження, а, отже, дають більше нащадків, ніж погано пристосовані особини. При цьому завдяки передаванню генетичної інформації (генетичному спадкуванню) нащадки успадковують від батьків основні їхні якості. Таким чином, нащадки сильних індивідуумів також будуть відносно добре пристосованими, а їхня частка в загальній масі особин буде зростати. Після зміни декількох десятків або сотень поколінь середня пристосованість особин даного виду помітно зростає [1, 2].

Еволюційна теорія стверджує, що кожен біологічний вид цілеспрямовано розвивається і змінюється для того, щоб якнайкраще пристосуватися до навколишнього середовища. У процесі еволюції багато видів комах і риб отримали захисне фарбування, їжак став невразливим завдяки голкам, людина стала власником дуже складної нервової системи. Можна сказати, що еволюція — це процес оптимізації усіх живих організмів. Розглянемо, якими засобами природа вирішує цю оптимізаційну задачу.

Щоб зробити зрозумілими принципи роботи генетичних алгоритмів, необхідно пояснити механізми генетичного спадкування в природі. Кожна клітина вміщує у собі всю генетичну інформацію особини. Ця інформація записана у вигляді набору дуже довгих молекул ДНК. Кожна молекула ДНК — це ланцюжок, який складається з молекул нуклеотидів (азотиста основа, пов'язана з вуглеводнем і залишком фосфорної кислоти) чотирьох типів, що позначаються A (аденін), T (тімін), C (цітозін) і G (гуанін). Інформацію надає порядок проходження нуклеотидів у ДНК. Таким чином, генетичний код індивідуума — це просто дуже довгий ряд символів, де використовуються

всього 4 букви. Уся сукупність генетичних ознак людини кодується за допомогою приблизно 60 тис. генів, довжина яких складає більше 90 млн. нуклеотидів. У клітині кожна молекула ДНК оточена оболонкою — таке утворення називається хромосомою. Кожна вроджена якість особини (колір ока, спадкові хвороби, тип волосся тощо) кодується визначеною частиною хромосоми, яка називається геном цієї властивості. Наприклад, ген кольору ока містить інформацію, яка кодує визначений колір очей. Різні значення гена називаються його алелями [1].

Зазначимо, що загалом розрізняють два види клітин: статеві і соматичні. У кожній соматичній клітині людини є 46 хромосом, що формують 23 пари, в яких одна з хромосом отримана від батька, а друга — від матері. Парні хромосоми відповідають за однакові ознаки — наприклад, батьківська хромосома може містити ген чорного кольору ока, а парна їй материнська — ген блакитного кольору. При розмноженні тварин відбувається злиття двох батьківських статевих клітин, а їхні ДНК взаємодіючи, утворюють ДНК нащадка. Основний спосіб взаємодії схрещування — кроссовер або кроссінговер. При кроссовері ДНК предків розділяється на частини, а потім потім відбувається обмін певними ділянками.

Важливий фактор, який впливає на спадковість, — це мутації, що виражаються в зміні деяких ділянок ДНК. Мутації випадкові і можуть бути викликані різними зовнішніми факторами. У результаті можуть змінитися деякі гени в статевих клітинах одного з батьків. Змінений ген може передатися нащадкові і проявитися у виді спадкової хвороби або в інших нових властивостях. Якщо ці нові властивості корисні, вони, швидше за все, збережуться для даного виду — при цьому відбудеться стрибкоподібне підвищення його пристосованості. Вважається, що саме мутації є причиною появи нових біологічних видів, а кроссовер визначає мінливість усередині виду (наприклад генетичні розходження між людьми).

Отже, основний закон спадкування інтуїтивно зрозумілий кожному — він полягає в тому, що нащадки схожі на батьків. Зокрема нащадки більш

пристосованих батьків будуть, швидше за все, одними з найбільш пристосованих у своєму поколінні [1-3].

## 1.2 Еволюційне моделювання

Системи моделювання еволюційних процесів поділяють на дві категорії:

1. Системи, в яких використовуються еволюційні принципи. Вони описують математичні моделі функціональної оптимізації. До них відносять еволюційні алгоритми, а саме: еволюційне програмування, генетичні алгоритми, генетичне програмування, еволюційні стратегії.

2. Системи, які є більш реалістичними з біологічної точки зору, але не мають широкого прикладного застосування – "штучне життя" (*artificial life*). За їх допомогою моделюють складні дії, наближені до реальних процесів.

Генетичні алгоритми разом з еволюційною стратегією й еволюційним програмуванням представляють три головних напрямки розвитку еволюційного моделювання (*simulated evolution*) [4-7]. Усі три представлених методи обчислень (рисунок 1.1) також поєднуються під загальною назвою еволюційні алгоритми (*evolutionary algorithms*).

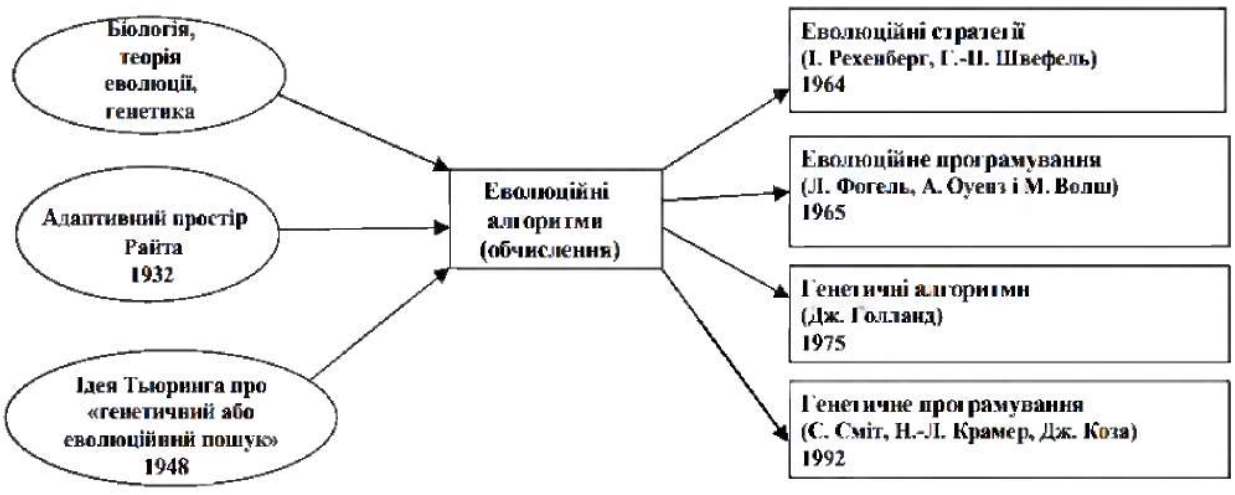


Рисунок 1.1 – Методи еволюційних обчислень

Еволюційне програмування (*evolutionary programming*) винайдене Л. Фогелем у 1960-1965 роках. Він помітив можливість застосування альтернативного підходу до проблем штучного інтелекту — моделювання не кінцевого результату еволюції, а самого процесу еволюції як засобу ”відпрацювання розумної поведінки” і можливостей передбачення різних явищ у середовищі. Ним виконано низку експериментів, у яких скінченні автомати представляли собою особин у популяції розв’язків задачі. Для даних автоматів було передбачено використання символів у цифрових послідовностях, які, еволюціонуючи, ставали більш придатними для розв’язування даної задачі [8].

Генетичний алгоритм (*genetic algorithm*) — це модель еволюції в природі (евристика), яка реалізована у виді комп'ютерної програми, що дозволяє розв’язати певну задачу за поліноміальний час. У її основі лежить використання еволюційних принципів (аналог механізму генетичного спадкування і природного відбору) для пошуку оптимального розв'язання задачі. При цьому зберігається біологічна термінологія в спрощеному виді [9].

Від традиційних методів оптимізації генетичні алгоритми вирізняються тим, що [10-12]:

– використовують не значення параметрів задачі, а їх закодовані форми (кодування параметрів);

– здійснюють пошук розв’язку не з єдиної початкової точки, а відштовхуючись від деякої популяції;

– використовують тільки цільову функцію, а не її похідні чи інші допоміжні дані;

– застосовують ймовірнісні, а не детерміновані правила вибору (рандомізація операцій);

– застосовуються до задач, які розв’язуються тільки методом грубої сили (повним перебором).

Генетичне програмування (*genetic programming*) було розроблене Дж. Козою у США у 90-их роках [13]. При використанні цього методу популяція складається з закодованих відповідним чином програм, які піддаються впливові генетичних операторів схрещування і мутації для знаходження оптимального розв'язку — програми, що щонайкраще вирішує поставлену задачу. Особливістю цього виду еволюційних алгоритмів є використання нелінійних хромосом (графи, дерева). Здебільшого використовують або мутацію (заміна випадковим деревом деякого піддерева розв'язку), або схрещування (обмін батьківських хромосом піддеревами). Існує можливість отримувати більш гнучкі зміни у розв'язках, а також оперувати хромосомами різного розміру (деревами з різною глибиною).

Еволюційні стратегії (*evolutionary strategy*) застосовують для неперервних значень, які є типовішими для експериментальних задач. Технологія еволюційних стратегій була винайдена І. Рехенбергом, а згодом розвинута Г.-П. Швефелем та іншими вченими [14-16].

Перший контакт між науковими колективами, що розвивали еволюційні стратегії й еволюційне програмування, відбувся на початку 1992 року, безпосередньо перед першою міжнародною конференцією, присвяченою еволюційному програмуванню. Ці методи розвивалися незалежно впродовж 30 років. Вони роблять акцент на адаптацію і різноманітність способів передавання властивостей від батька до нащадків у наступних поколіннях [17].

Розглянуті методи відрізняються здебільшого представленнями можливих розв'язків задачі. В еволюційних стратегіях набори хромосом представляють векторами дійсних чисел, в генетичних алгоритмах — векторами двійкових чисел, в еволюційному програмуванні — скінченними автоматами, а в генетичному програмуванні — деревами (списками).

Для позначення різноманітних алгоритмів, заснованих на еволюційному підході, також застосовується поняття еволюційних програм (*evolution*

*programs*). Цей термін поєднує як генетичні алгоритми, еволюційні стратегії й еволюційне програмування, так і генетичне програмування, а також інші аналогічні методи [17].

На сьогодні існує низка прикладних програмних продуктів, в яких реалізовано інструментарій еволюційних алгоритмів. Залежно від сфери використання, ступеня автономності та призначення, їх можна класифікувати наступним чином [18]:

– спеціалізоване програмне забезпечення, яке створене для розв'язання вузького кола прикладних задач. Нескладні для користування, але обмежені для використання;

– додатки до математичних та аналітичних пакетів, через які надаються можливості розв'язання різноманітних задач, але більшість математичних пакетів не є пропріетарними, а їх використання потребує високого рівня знань та навичок;

– фреймворки представляють безкоштовне програмне забезпечення. За їх допомогою користувач, який має певні знання і навички у програмуванні, може розробляти власні програмні засоби на основі існуючого коду.

Слід зазначити, що різні види еволюційних алгоритмів у чистому вигляді застосовують досить рідко, а дедалі частіше використовуються гібридні еволюційні алгоритми, які дозволяють пришвидшити розв'язання конкретної задачі і покращити результати.

## 1.3 Постановка задачі оптимізації функції

Будемо розв'язувати задачу оптимізації нелінійної функції, тобто здійснимо пошук її екстремумів. Нехай дана деяка цільова функція $F$, яка залежить від декількох змінних $x_i$, і потрібно знайти такі значення аргументів, при яких функція набуває максимуму або мінімуму (екстремуму). Задачі такого

роду називаються задачами оптимізації, і вони дуже часто зустрічаються на практиці. Математичну постановку задачі оптимізації можна задати наступним чином: потрібно знайти такий набір аргументів $x_i$ із множини допустимих рішень $D$, при якому забезпечується екстремальне значення цільової функції $F(x_1,...,x_n)$, тобто [19]

$$F(x_1,...,x_n) \to ext\ (\min,\ \max),\ x_i \in D,\ i = 1,...,n. \qquad (1.1)$$

Один з найбільш наочних прикладів — задача розподілу інвестицій. У цій задачі змінними є обсяги інвестицій у кожний проект, а функцією, яку потрібно максимізувати — сумарний дохід інвестора.

## 1.4 Висновки

У даному розділі розглянуто передумови виникнення еволюційних обчислень та еволюційного моделювання. Представлено основи еволюційної теорії, яка ґрунтується на поняттях природного відбору та генетичного спадкування. Генетичний алгоритм є важливим напрямком розвитку еволюційного моделювання і представляє собою евристику, яка дозволяє розв'язати оптимізаційну задачу за поліноміальний час. Еволюційні алгоритми широко застосовують у сучасній теорії оптимізації. У даній роботі досліджується ефективність розв'язання задач неперервної оптимізації генетичними алгоритмами.

# 2 РОЗРОБКА МАТЕМАТИЧНОЇ МОДЕЛІ ОСТРІВНОГО ГЕНЕТИЧНОГО АЛГОРИТМУ

## 2.1 Механізм роботи генетичного алгоритму

Типова практична задача, як правило, мультимодальна (має більше одного екстремуму) і багатовимірна (містить багато змінних). Для таких задач не існує жодного універсального методу, що дозволяв би досить швидко знайти абсолютно точне розв'язання. Крім того, для мультимодальних функцій існує серйозна проблема передчасної збіжності до локальних екстремумів. Однак, комбінуючи переборний і градієнтний методи, можна сподіватися отримати наближений розв'язок, точність якого буде зростати при збільшенні часу обчислень. Генетичний алгоритм представляє собою саме такий комбінований метод пошуку екстремуму.

Отже, якщо на деякій множині задана складна функція $F$, то генетичний алгоритм — це програма, яка за розумний час знаходить точку, де значення функції досить близьке до екстремального. Вибираючи прийнятний час обчислень, можна отримати одне з кращих рішень, що взагалі можливо одержати за цей час [20-25].

Принципи роботи алгоритму [1]:

1. Природний відбір (Ч. Дарвін, 1859 рік). Виживають найбільш пристосовані до навколишнього середовища особини популяції.

2. Наслідування (Г.І. Мендель, 1865 рік). Хромосоми нащадків складаються з частин хромосом їх батьків.

3. Мутація (Г. де Фріз, 1900 рік). Суттєві зміни властивостей нащадків; формування властивостей, які відсутні у батьків. Подібний механізм використовується для підвищення різноманітності (мінливості) особин у популяції.

Тобто під час роботи з генетичними алгоритмами зберігається біологічна термінологія в спрощеному вигляді (таблиця 2.1) [26].

Таблиця 2.1 – Термінологія генетичного алгоритму

| Природа | Комп'ютер |
|---|---|
| Популяція | Множина особин $N$ |
| Хромосома | Вектор із нулів та одиниць довжиною $l$; упорядкована послідовність генів |
| Індивідуум, генотип | Набір хромосом — варіант розв'язку задачі (значення аргументів функції) |
| Фенотип | Набір значень, які відповідають даному генотипу. Декодована структура або множина параметрів задачі (значення $F$, точка простору пошуку) |
| Ген | Атомарний елемент генотипу або хромосоми. Кожна позиція (біт) заданого вектору $l$ |
| Локус, позиція | Місце розміщення даного гена в хромосомі |
| Алель | Значення конкретного гена (0 або 1) |
| Функція пристосованості, фітнес-функція | Міра пристосованості особини у популяції (проміжне значення функції з простору рішень) |
| Кросовер, кросінговер | Схрещування. Операція, під час якої хромосоми обмінюються частинами |
| Мутація | Випадкова зміна позицій в хромосомі |
| Інверсія | Перестановка фрагментів хромосом |

На початку роботи алгоритму випадковим чином ініціалізується популяція (часто застосовують код Грея), яка має такі характеристики, як: чисельність $v$, кількість хромосом $N$ у кожної особини і їх розрядність $l$. Особини популяції оцінюються відповідно до вибраного критерію, і в

результаті оцінки визначається їх пристосованість $F$. Наприклад, у теорії керування функція пристосованості може приймати вид функції похибки, а в теорії ігор – вартісної функції. На кожній ітерації генетичного алгоритму пристосованість кожної особини даної популяції оцінюється за допомогою фітнес-функції, і на цій основі створюється наступна популяція особин, які складають множину потенційних рішень задачі оптимізації. Чим вища пристосованість особини, тим більша ймовірність того, що вона візьме участь в схрещуванні, під час якого відбувається ”обмін генетичною інформацією”, тобто відповідні хромосоми обмінюються генами та мутують. Механізми схрещування і мутації в певному сенсі реалізують перебір рішень, а підбір кращих із них виконує градієнтний спуск. Отримані в результаті схрещування нащадки формують нове покоління, для якого все повторюється спочатку [27].

Алгоритм припиняє роботу, якщо знайдено рішення, минув встановлений час роботи або популяція тривалий час не прогресує, тобто рядки популяції знаходяться в області деякого екстремуму і майже однакові. Після закінчення роботи найбільш пристосована особина популяції, точніше її гени, будуть представляти рішення задачі. Загалом роботу генетичних алгоритмів можна описати схемою на рисунку 2.1 [28, 29].


Рисунок 2.1 — Етапи обчислень в генетичному алгоритмі

Генетичні оператори (кроссовер, мутація, інверсія) необхідні, щоб застосувати принципи спадковості і мінливості до віртуальної популяції. Такі оператори не обов'язково застосовуються до всіх особин, що вносить додатковий елемент невизначеності в процес пошуку рішення. У даному випадку, невизначеність не має на увазі негативний фактор, а є певним

”степенем свободи” генетичного алгоритму. Слід зазначити, що існує багато генетичних операторів, але у даній роботі опишемо найбільш необхідні [15].

Оператор кроссовера (*crossover operator*) моделює процес схрещування і є основним генетичним оператором, за рахунок якого виконується обмін генетичним матеріалом між особинами. Ймовірність застосування кроссовера $p_{кр} \geq 50\%$. Нехай є дві батьківські особи з хромосомами $X = \{x_i,\ i = 1,\ldots,n\}$ і $Y = \{y_i,\ i = 1,\ldots,n\}$. Випадковим чином визначається точка всередині хромосоми — точка розриву або кроссоверу (*crossover point*), у якій обидві хромосоми діляться на дві частини і обмінюються ними (рисунок 2.2). Даний тип кроссоверу називається одноточковим, оскільки при ньому батьківські хромосоми розрізаються тільки в одній випадковій точці. Також існує *n-точковий* оператор кроссоверу ($n \geq 2$) [7, 9, 21, 30].

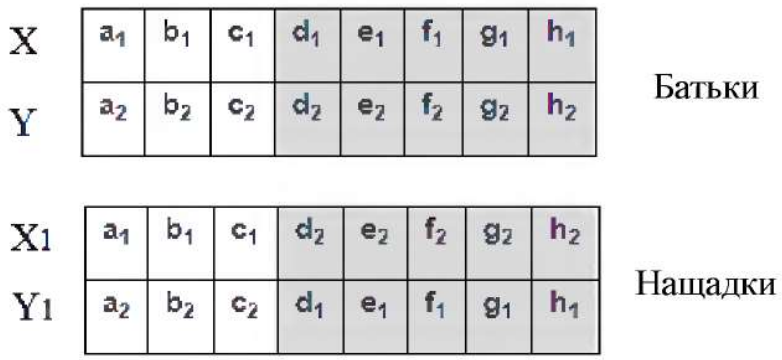


Рисунок 2.2 — Приклад одноточкового кросоверу

Виділяють також однорідний кроссовер. Його особливість полягає в тому, що значення кожного біта в хромосомі нащадка визначається випадковим чином з відповідних бітів батьків. Для цього вводиться величина ймовірності $0 < p_0 < 1$, і якщо випадкове число більше $p_0$, то на позицію гена $i$ першого нащадка потрапляє $i$-й біт першого батька, а на $i$-ту позицію другого — $i$-ий біт другого батька. У протилежному випадку до першого нащадка потрапляє біт другого батька, а до другого – першого. Така операція проводиться для всіх бітів хромосоми [9, 21].

Оператор мутації (*mutation operator*) необхідний для «вибивання» популяції з локального екстремуму, крім того, він захищає від передчасної збіжності. Це досягається за рахунок того, що інвертується випадково вибраний біт у хромосомі. Ймовірність мутації $p_{мут}$ значно менша ймовірності кроссовера і рідко перевищує 10%. Серед рекомендацій з вибору ймовірності мутації нерідко можна зустріти варіанти $1/l$ або $1/N$, де $l$ — довжина хромосоми, $N$ — розмір популяції. Необхідно відзначити, що деякі автори вважають, що оператор мутації є основним пошуковим оператором [9, 21].

Так само як і кроссовер, мутація проводиться не тільки по одній випадковій точці (рисунок 2.3). Можна вибирати деяку кількість точок у хромосомі, причому їхня кількість також може бути визначена випадковим числом. Крім того, можна змінювати відразу деяку групу точок.

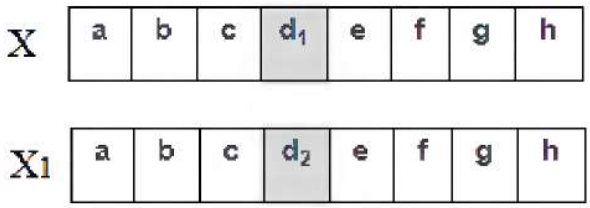


Рисунок 2.3 — Приклад одноточкової мутації

Після схрещування особин необхідно вирішити, які з них ввійдуть у наступне покоління, а які — ні, і що робити з їх предками. Можна виділити дві стратегії відбору членів нової популяції [22, 29]:

1. Витіснення. Нові особини (нащадки) займають місця своїх батьків. Після чого настає наступний етап, у якому нащадки оцінюються, відбираються, дають потомство і поступаються йому місцем.

2. Елітизм. Створюється проміжна популяція, яка містить у собі як батьків, так і їхніх нащадків. Члени цієї популяції оцінюються, а потім з них вибираються найкращі, які ввійдуть у наступне покоління.

У класичному описі генетичного алгоритму мається на увазі створення популяції нащадків і витіснення ними батьківських особин. Такий підхід

досить хороший, але не особливо ефективний, тому що нащадки, отримані в результаті генетичних перетворень, можуть бути гірші батьків. Тому існує стратегія елітного відбору (елітизму), яка виправляє дане явище. Його можна розділити на два класи [22]:

1. Конкуренція. Батьківські особини «змагаються» з нащадками, а переможці (або переможець) переходять у наступне покоління.

Іншими словами нащадки після створення мають рівні права з батьками на те, щоб перейти в нове покоління, а визначальну роль відіграє пристосованість особини, а не її положення на генеалогічному дереві.

Загалом можна виконати процедуру відбору за змаганням особин. При глобальному змаганні спочатку створюються усі нащадки, які потім змагаються на загальних підставах з усіма батьками, а ті, хто виявляться кращими, переходять у нове покоління. При локальному змаганні нащадки змагаються тільки зі своїми батьками. Тут проблеми родини вирішуються всередині родини.

З погляду еволюційних стратегій конкурентний підхід відповідає *mu+ lambda* стратегії або «плюс-стратегії» (*plus strategy*), де *mu* — кількість батьківських особин, а *lambda* — кількість створених нащадків.

2. Без конкуренції. У цьому випадку частина батьківської підпопуляції (випадкова або визначена за заданим правилом) переходить у нове покоління без яких-небудь заперечень з боку електорату. Тобто, якщо навіть усі нащадки будуть кращі будь-якого батька, то все одно певна частина батьківської підпопуляції неминуче буде присутня у новому поколінні.

В еволюційних стратегіях неконкурентний підхід відповідає (*mu*, *lambda*) стратегії або «кома-стратегії» (*comma strategy*), де зміст змінних *mu* і *lambda* залишається колишнім.

Очевидно, що при елітній стратегії відбору повільніше оновлюється генетична інформація в популяції, оскільки частина особин, точніше їх хромосоми, залишаються незмінними при зміні поколінь. Якщо популяція «наближається» до глобального оптимуму (знаходиться на невеликій відстані

від нього), то різкі зміни особин (більшість з яких мають високу пристосованість) небажані, тому що можуть погіршити відповідні рішення. Тому, у даному випадку, елітна стратегія, як спосіб збереження хороших особин, необхідна. Якщо ж еволюційний пошук знаходиться на самому початку, то необхідно активно досліджувати простір пошуку, щоб виділити в ньому потенційні області, при влученні в які, популяція буде мати високу пристосованість. Тому на початкових етапах треба використовувати елітну стратегію більш обережно.

Тобто, якщо використовується елітизм, іншими словами, більш активно використовуються вже знайдені гарні особини, то необхідні міри, які будуть компенсувати вплив елітизму, наприклад збільшення ймовірності мутації, розміру популяції; заборона на існування особин-близнюків у популяції. У протилежному випадку істотно збільшується ризик передчасної збіжності. Але у цілому використання елітної стратегії дозволяє істотно поліпшити результати роботи генетичного алгоритму.

Зазвичай при відборі конкурентних особин-батьків (операція селекції) використовують пропорціональний відбір, заснований на методі колеса рулетки (*roulette wheel selection*). У ньому ймовірність $p(X_i)$ вибору хромосоми $X_i$ виявляється пропорційною значенням її функції пристосованості. Кожній хромосомі $X_i$ $(i = 1,2,...,N)$ відповідає сектор колеса рулетки $v(X_i)$, який визначено у відсотках за формулою

$$v(X_i) = p(X_i) \cdot 100\% = \frac{F(X_i)}{\sum_{i=1}^{N} F(X_i)} \cdot 100\%, \qquad (2.1)$$

де $F(X_i)$ — значення фітнес-функції хромосоми $X_i$; $p(X_i)$ — ймовірність селекції (вибору) хромосоми $X_i$.

Вибір хромосоми можна ототожнити з вибором числа з інтервалу [$a$; $b$], де $a$ та $b$ позначають відповідно початок і кінець фрагмента кола, відповідного

сектору колеса рулетки. Отже, у цьому випадку вибір зводиться до вибору числа з інтервалу [0, 100], яке відповідає конкретній точці на колі колеса.

Ще одним підходом є лінійне ранжування, в якому використовується тільки номер особини в упорядкованому списку, тоді ймовірність її вибору можна описати формулою

$$p(X_i) = \frac{1}{N} \cdot \left( a - (a-b) \cdot \frac{i-1}{N-1} \right) \cdot 100\%, \qquad (2.2)$$

де $1 \le a \le 2$ — вибирається випадково; $b = 2 - a$; $i$ — номер особини в упорядкованому списку.

Також можна застосовувати рівномірне ранжування, яке задається наступною формулою

$$p(X_i) = \begin{cases} \frac{1}{\mu}, & якщо\ 1 \le i \le \mu; \\ 0, & якщо\ \mu < i \le N, \end{cases} \qquad (2.3)$$

де $\mu \le N$ — параметр методу.

Іншим підходом є метод Больцмана, який базується на алгоритмі імітації відпалу. Для керування пошуком вводиться штучна змінна температури $T$, яка, починаючи з деякого великого значення, поступово зменшується за певним законом і змінює ймовірність відбору особин. Зі зменшенням температури $T$ різниця значень $p(X_i)$ між найкращими та найгіршими особинами збільшується. Ймовірність вибору можна описати виразом

$$p = \frac{1}{1 + \exp\left( \frac{F(X_i) - F(X_j)}{T_t} \right)} \cdot 100\%, \qquad (2.4)$$

де $f(X_i), f(X_j)$ – значення фітнес-функції особин $i$ та $j$ (їх номера вибирають випадковим чином); $T_{t+1} = k \cdot T_t$; $t$ – номер ітерації; $k = 0{,}8 \ldots 0{,}99$.

Якщо значення $p$ буде більшим за $rand(0,1)$, то в нову популяції потрапить особина $i$ (більше значення функції), інакше – особина $j$.

Батьківські пари можна вибирати такими способами [26]:

1. Панміксія — члени популяції, які створюють батьківські пари, вибираються випадковим чином із усієї популяції, при чому будь-який член популяції може стати учасником декількох пар.

2. Селективний — батьками можуть стати лише ті особини, значення пристосованості яких не менше середнього значення пристосованості по популяції.

3. Інбрідінг — перший член пари $X$ вибирається випадковим чином, а другим $Y$ з більшою ймовірністю буде максимально близький до нього за вістанню Хеммінга (по алелям), тобто

$$d_{ij} = W(X \oplus Y) = \sum_{k=1}^{l} | g_{ik}^{X} - g_{jk}^{Y} | \to \min, \qquad (2.5)$$

де $W$ — вагова функція; $g$ — алель.

4. Аутбрідінг — пари формуються аналогічно, але із максимально далеких членів популяції, тобто $d_{ij} \to \max$.

## 2.2 Теорема Холланда

Для того щоб краще зрозуміти функціонування генетичного алгоритму, будемо використовувати поняття шаблону (схема, шіма), яке було введене Холландом для аналізу роботи генетичних алгоритмів. Зокрема розглядаються процеси конструювання і руйнування визначеного шаблону впродовж розвитку популяції (динамічна схема) [7, 22]. Шаблоном $H$ називається рядок виду

$$H = (a_1, a_2, ..., a_i, ..., a_n), \text{ де } a_i \in \{0;1;*\}, \qquad (2.6)$$

Символом «*» у деякому розряді позначається той елемент, який там може бути, тобто 0 або 1. Наприклад, для двох бінарних рядків 111000111000 і 110011001100 шаблон буде виглядати як 11*0****1*00. За допомогою шаблонів можна виділяти загальні ділянки двійкових рядків і маскувати розходження. Маючи в складі схем $m$ символів «*», можна закодувати (узагальнити) $2^m$ двійкових рядків. Наприклад, шаблон 01*0*1 описує набір рядків {010001, 010011, 011001, 011011}.

Порядок шаблону $o(H)$ (*schema order*) — це характеристика шаблону, яка дорівнює кількості фіксованих позицій 0 і 1 в рядку. Визначаючою довжиною шаблону $\Delta(H)$ (*schema defining length*) називається відстань між двома крайніми символами 0 і / або 1. Для шаблонів 01*0*1 і **0**1* порядки дорівнюють 4 і 2, а визначаючі довжини — 5 і 3 відповідно.

Хромосома, по суті, є двійковим рядком. У той же час особині, якій належить хромосома, поставлена у відповідність величина, що характеризує її пристосованість. Оскільки шаблон є узагальненням декількох бінарних рядків (хромосом), то можна говорити, що особини, які володіють хромосомами, що відповідають одному шаблону, більш пристосовані, а особини з хромосомами, які відповідають іншій схемі — менш пристосовані.

Можна сказати, що суть роботи генетичного алгоритму полягає в пошуку двійкового рядка визначеного виду з усієї множини бінарних рядків. Простір пошуку складає $2^l$ рядків, а його розмірність дорівнює $l$, де $l$ — довжина хромосоми. Схема відповідає деякій гіперплощині в цьому просторі. Дане твердження можна проілюструвати в такий спосіб. Нехай розрядність хромосоми дорівнює 3, тоді усього можна закодувати $2^3 = 8$ рядків. Представимо куб у 3-вимірному просторі. Позначимо вершини цього куба 3-розрядними бінарними рядками так, щоб мітки сусідніх вершин відрізнялися рівно на один розряд, причому вершина з міткою 000 знаходилася б на початку координат. Варіант позначення зображено на рисунку 2.4. Якщо взяти шаблон виду **0, то він опише ліву грань куба, а схема *10 — верхнє ребро цієї грані. Очевидно, що шаблон *** відповідає всьому просторові.

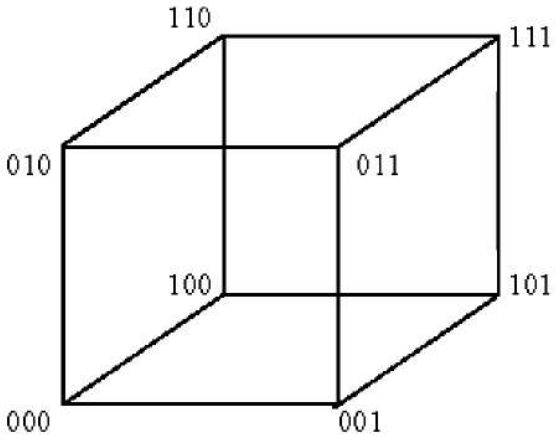


Рисунок 2.4 — Простір пошуку у вигляді 3-вимірного куба

Якщо взяти двійкові рядки довжиною 4 розряди, то розбиття простору пошуку схемами можна зобразити на прикладі 4-вимірного куба або гіперкуба (рисунок 2.5). Тут схемі *1** відповідає гіперплощина, яка включає задні грані зовнішнього і внутрішнього куба, а схемі **10 — гіперплощина з верхніми ребрами лівих граней обох кубів.

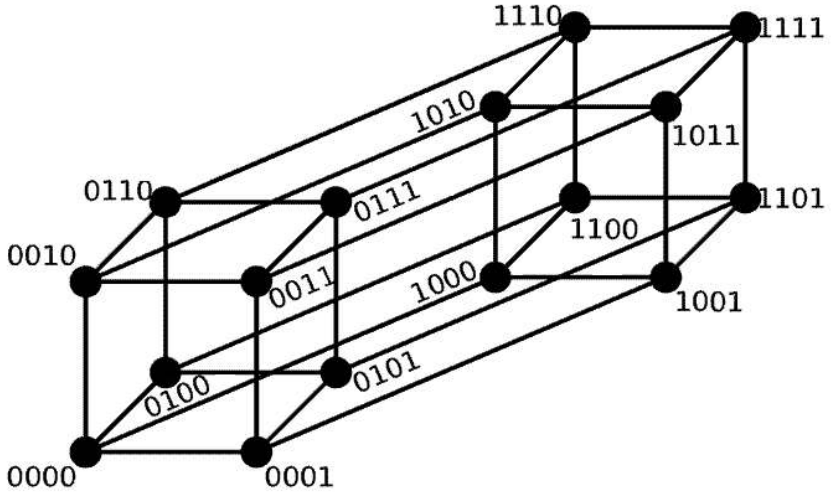


Рисунок 2.5 — Простір пошуку у вигляді гіперкуба

Генетичний алгоритм базується на принципі трансформації найбільш пристосованих особин (їхніх хромосом). Нехай $P(0)$ означає вихідну популяцію особин, а $P(k)$ — поточну популяцію на $k$-ій ітерації алгоритму. З кожної популяції $P(k)$ методом селекції вибираються хромосоми з найбільшою

пристосованістю, які включаються в батьківський пул (*mating pool*) $M(k)$. Далі, у результаті об'єднання особин з популяції $M(k)$ у батьківські пари і виконання операції схрещування з ймовірністю $p_{кр}$, а також операції мутації з ймовірністю $p_{мут}$ формується нова популяція $P(k+i)$, у яку входять нащадки особин з популяції $M(k)$. Отже, для будь-якого шаблону, який представляє гарне рішення, було б бажаним, щоб кількість хромосом, що йому відповідає, зростала зі збільшенням кількості ітерацій $k$.

Теорема шаблонів (*The Schema Theorem*), сформульована Холландом в 1975 р. та відкоригована у 1992 р., показує, як змінюється частка представників шаблону в популяції [2]. Шаблони малого порядку, малої визначаючої довжини, з пристосованістю вище середньої формують показово зростаючу кількість своїх представників в подальших поколіннях генетичного алгоритму. Дану теорему можна записати наступною нерівністю

$$P(H,t+1) \geq P(H,t) \cdot \frac{F(H,t)}{\overline{F(t)}} \cdot \left(1 - p_{кр} \cdot \frac{\Delta(H)}{l-1}\right) \cdot \left(1 - p_{мут}\right)^{o(H)}, \qquad (2.7)$$

де $H$ — шаблон; $P(H, t)$ — частка представників $H$ в $t$-ому поколінні; $F(H, t)$ — значення пристосованості $H$ в $t$-ому поколінні; $\overline{F(t)}$ — середнє значення пристосованості в $t$-ому поколінні; $p_{кр}$ і $p_{мут}$ — ймовірність кроссоверу та мутації; $\Delta(H)$ — визначаюча довжина $H$; $l$ — довжина хромосоми; $o(H)$ — порядок $H$.

Слід зауважити, що теорема справедлива тільки для класичного алгоритму. Важливим питанням стає кодування, яке повинно забезпечувати побудову таких шаблонів. Непрямим результатом теореми можна вважати наступну гіпотезу Холланда (про будівельні блоки): генетичний алгоритм прагне досягти близького до оптимального результату за рахунок комбінування хороших шаблонів (малі порядок та визначаюча довжина, пристосованість вище середньої), які названо будівельними блоками (*building blocks*).

## 2.3 Моделі генетичних алгоритмів

Для реалізації алгоритму існує ряд відомих моделей [17, 22]:

1. Канонічна (класична) модель (*canonical*). Вона була запропонована Холландом у його фундаментальній роботі [2] і має наступні параметри:

– фіксований розмір популяції;

– фіксована розрядність генів;

– пропорційний відбір;

– особини для схрещування вибираються випадковим чином;

– одноточкові кроссовер і мутація;

– наступне покоління формується з нащадків поточного покоління без елітизму. Нащадки займають місця своїх батьків.

2. Генітор (*genitor*). Дана модель, розроблена Уитли використовує специфічну стратегію відбору. На кожному кроці тільки одна пара випадкових батьків створює тільки одного нащадка, який замінює одну з найгірших особин популяції. Таким чином, на кожному кроці в популяції оновлюється тільки одна особина. Параметри моделі [29]:

– фіксований розмір популяції;

– фіксована розрядність генів;

– особини для схрещування вибираються випадковим чином;

– обмежень на тип кроссовера і мутації немає;

– один нащадок займає місце найменш пристосованої особини.

3. Гібридна модель (*hybrid*), яка розроблена Девісом. Її використання дозволяє об'єднати переваги генетичного алгоритму і класичних методів оптимізації. Справа в тому, що генетичні алгоритми є робастними, тобто вони дозволяють знаходити гарне рішення, але знаходження оптимального рішення найчастіше виявляється набагато більш важкою задачею в силу стохастичності принципів роботи алгоритму. Тому виникла ідея використовувати генетичний алгоритм на початковому етапі для ефективного звуження простору пошуку

навколо глобального екстремуму, а потім, узявши кращу особину, застосувати один з класичних методів оптимізації. Характеристики моделі [22]:

– фіксований розмір популяції;

– фіксована розрядність генів;

– будь-які комбінації стратегій відбору і формування наступного покоління;

– обмежень на тип кроссовера і мутації немає;

– генетичний алгоритм застосовується на початковому етапі, а потім у роботу включається класичний метод оптимізації.

4. Острівна або паралельна модель (*island*). Уявимо собі наступну ситуацію. У деякому океані є група близько розташованих островів, на яких живуть популяції особин одного виду. Ці популяції розвиваються незалежно, і тільки зрідка відбувається обмін представниками між популяціями. Острівна модель використовує описаний принцип для пошуку рішення. Крім того, вона дозволяє використати концепцію паралелізму. Дана модель генетичного алгоритму має наступні властивості [22]:

– наявність декількох популяцій (однакового фіксованого розміру);

– фіксована розрядність генів;

– будь-які комбінації стратегій відбору і формування наступного покоління в кожній популяції. Можна зробити так, що в різних популяціях будуть використовуватися різні комбінації стратегій, хоча навіть один варіант дає різноманітні рішення на різних «островах»;

– обмежень на тип кроссовера і мутації немає;

– випадковий обмін особинами між «островами». Якщо міграція буде занадто активною, то особливості острівної моделі будуть згладжені, тобто вона буде не дуже сильно відрізнятися від моделей без паралелізму.

5. CHC (*Cross generational elitist selection*). Модель розроблена Ешлеманом. Для нового покоління вибираються *n* кращих різних особин серед батьків і дітей, тобто дублювання рядків не допускається. Для схрещування всі особини розбиваються на пари, але схрещуються лише ті пари, між якими

відстань Хеммінга більша деякого порогового значення. При схрещуванні використовується різновид однорідного кросовера *HUX* (*Half Uniform Crossover*), тобто кожному нащадку переходить рівно половина бітів кожного з батьків. Даний алгоритм досить швидко сходиться із-за того, що в ньому немає мутацій, використовуються популяції невеликого розміру, а відбір особин в наступне покоління ведеться і між батьківськими особинами, і між їхніми нащадками. В силу цього після знаходження деякого рішення алгоритм перезапускається, при цьому найкраща особина копіюється в нову популяцію, а інші — піддаються сильній мутації (мутує приблизно третина бітів у хромосомі), після чого пошук повторюється.

Модель має наступні властивості [29]:

– фіксований розмір (невеликий) популяції;

– фіксована розрядність генів;

– особини для схрещування розбиваються на пари та схрещуються при наявності вагомих відмінностей;

– використовується половинний однорідний кроссовер та макромутація під час перезапуску;

– перезапуск алгоритму після знаходження рішення;

– наступне покоління формується з найкращих батьків та нащадків.

## 2.4 Особливості острівної моделі

Оскільки генетичні алгоритми є робастними, тобто дозволяють знаходити гарне рішення, але знаходження оптимального рішення найчастіше виявляється набагато більш важкою задачею в силу стохастичності принципів роботи алгоритму, то для отримання покращеного рішення будемо застосовувати острівну модель, що дозволяє використати принципи паралелізму, тобто

$$f(x^*) = F(F_{ген1}(\{x_i\}), F_{ген2}(\{x_i\}), \ldots, F_{ген\,n}(\{x_i\})). \qquad (2.8)$$

Наступний запуск алгоритму базується на досвіді попереднього, тобто

$$f_{j+1}(x^*) = M(f_j(x^*)), \tag{2.9}$$

де $M$ – пам'ять алгоритму.

Даний процес повторюється певну кількість разів $s$ та виводиться найкраще рішення за всі запуски програми

$$f_{best}(x^*) = \min_{b=1,\ldots,s}(\{f_b(x^*)\}). \tag{2.10}$$

## 2.5 Висновки

У даному розділі наведено основні принципи роботи генетичного алгоритму: проаналізовано особливості застосування генетичних операторів, розглянуто стратегії відбору членів популяції, формування батьківських пар та ряд моделей генетичного алгоритму. Запропоновано використання острівної моделі, яка дозволяє використати концепцію паралелізму. Перевагами генетичних алгоритмів є те, що вони не вимагають ніякої інформації про поведінку функції; стійкі щодо потрапляння в локальний оптимум; придатні для вирішення масштабних проблем оптимізації; прості в реалізації тощо. У той же час за їх допомогою проблематично знайти точне значення глобального оптимуму; можуть використовувати дуже багато часу на обчислення; непросто налаштувати для знаходження всіх рішень задачі тощо.

У наступному розділі проведемо експериментальні дослідження ефективності роботи алгоритму у розробленому програмному додатку.

# 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

## 3.1 Опис роботи програми

Загалом можна виділити наступні етапи генетичного алгоритму [13, 30]:

1. Згенерувати $r$ островів.

2. Створити початкову популяцію розміром $N$ з довжиною хромосом $l$.

3. Обчислити функції пристосованості $F$ для особин популяцій.

4. Природний відбір: вибір індивидуумів з поточної популяції (селекція); схрещування і мутація; формування нового покоління.

5. Критерій зупинки: якщо виконуються умови закінчення обчислень, то перейти до пункту 6, інакше — повернутись до пункту 2.

6. Вивести результат роботи програми, тобто представити шукане рішення задачі. Кращим рішенням вважається хромосома з екстремальним значенням функції пристосованості. Для тестування необхідно провести $s$ запусків програми та вивести середнє значення $F$ за найкращими результатами.

Розроблене програмне забезпечення [21] надає можливість вибрати кількість поколінь ($v$); розмір популяції ($N$); розрядність генів ($l$); тип і ймовірність кроссоверу ($p_{кр}$) та мутації ($p_{мут}$) для особин популяції; метод селекції; принцип формування покоління; точність обчислень; кількість запусків для збору статистики ($s$); критерій зупинки; інтервали зміни параметрів (простір пошуку) та тестову функцію. Для задання математичних функцій у програмі *MATLAB* [31] використовуються файли з розширенням **.m* (*m-файли*). Результати роботи програми показують значення екстремуму функції $f_{min}$ та кількість знаходжень $f_{min}$ за $s$ запусків програми у відсотках.

## 3.2 Тестування ефективності

Усі тестові функції можуть мати різну кількість параметрів $n$. Тому має сенс запустити алгоритм для оптимізації деякої функції спочатку з невеликим $n$ (10 або 20), а потім з $n$ = 50, 100, 200,... Це дасть можливість перевірити масштабованість алгоритму. Ефективність роботи генетичного алгоритму прийнято оцінювати кількістю обчислень цільової функції: чим менше, тим краще. Оскільки генетичні алгоритми використовують стохастичність, то для того, щоб визначити, наскільки ефективний алгоритм потрібно запустити його на одній і тій же тестовій функції декілька разів. Після цього можна проаналізувати результат. Параметри запуску програми такі [13, 21]:

- кількість островів $r = 6$;
- максимальна кількість поколінь $v = 1000$;
- фіксований розмір популяції $N = 35$ на острів;
- фіксована довжина хромосоми $l = 10$;
- бінарне кодування даних;
- одноточковий кроссовер ($p_{кр} = 1$) та мутація ($p_{мут} = 0{,}5$);
- відбір особин на базі елітизму без конкуренції: дві ”елітні” особини;
- пропорціональний відбір батьків;
- формування пар батьків на основі панміксії;
- точність обчислень складає 0,001 (значення цільової функції, які менші даного, зараховувалися як знайдений глобальний оптимум);
- критерій зупинки обчислень: знаходження оптимуму; ліміт поколінь; часове обмеження;
- результат розрахований за $s = 30$ запусками алгоритму.

Окремі підпопуляції можна умовно прийняти за вершини деякого графа. У зв’язку з цим можна розглядати топологію графа острівного генетичного алгоритму. Найбільш поширеною топологією є повний граф (рисунок 3.1), при якій особи з будь-якої підпопуляції можуть мігрувати в будь-яку іншу підпопуляцію.

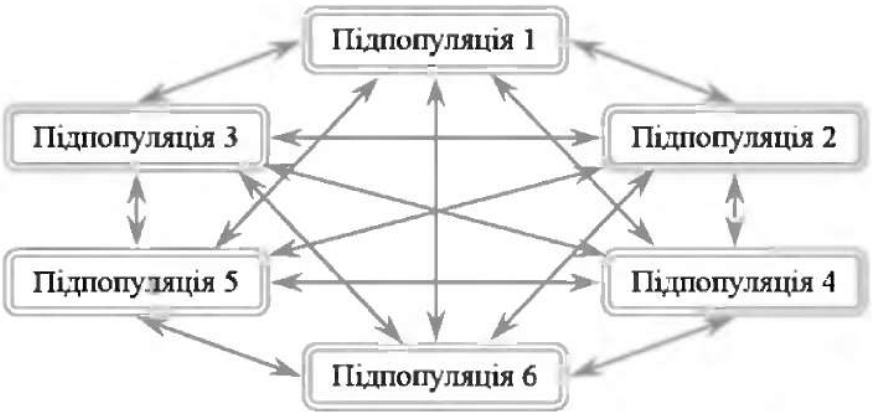


Рисунок 3.1 – Топологія острівної моделі

Нижче приводиться набір тестових функцій [32], для яких було проведено експерименти по знаходженню оптимуму генетичним алгоритмом.

1. Еліптичний параболоїд. Вважається простою функцією для будь-якого методу оптимізації. Один мінімум $F(x_i^*) = F(0,\ldots,0) = 0$ (рисунок 3.2).

$$F(x_1,\ldots,x_n) = \sum_{i=1}^{n} x_i^2,\ x_i \in [-5; 5]. \tag{3.1}$$

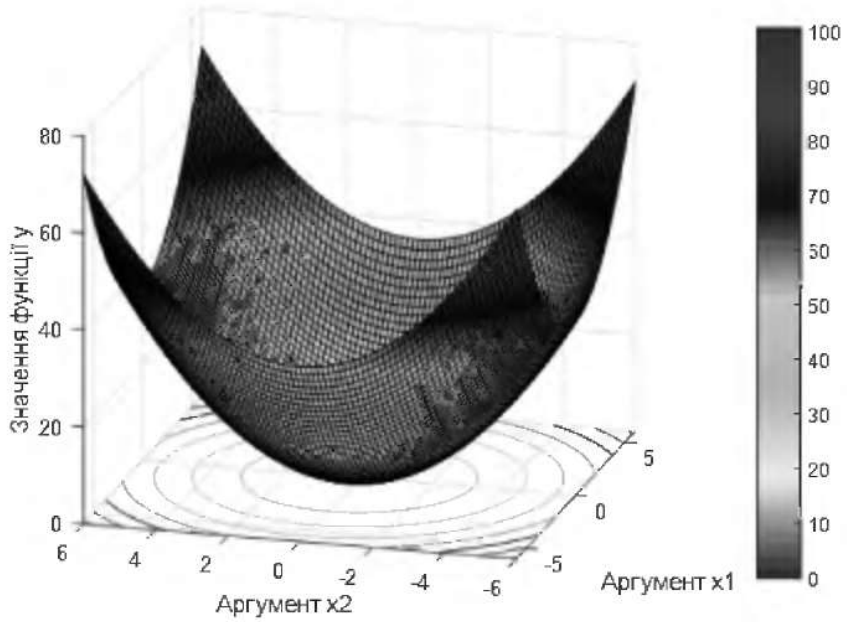


Рисунок 3.2 — Еліптичний параболоїд

Результати роботи програми:

— $n$ = 10, кількість знаходжень глобального мінімуму 95%, кількість обчислень цільової функції не більша 1225, максимальне значення функції 0,008225;

— $n$ = 30, кількість знаходжень глобального мінімуму 62%, кількість обчислень цільової функції не більша 1225, максимальне значення функції 0,117751;

— $n$ = 50, кількість знаходжень глобального мінімуму 63%, кількість обчислень цільової функції не більша 2430, максимальне значення функції 0,017391.

2. Функція Растрігіна. Вважається складною для оптимізації. Вона грунтується на функції (3.1) з додаванням косинусної модуляції для створення багатьох локальних мінімумів. Таким чином, ця тестова функція є мультимодальною, а положення мінімумів розподілені регулярно. Має мінімум $F(x_i^*) = F(0,\ldots,0) = 0$ (рисунок 3.3).

$$F(x_1,...,x_n) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)),\ x_i \in [-5; 5]. \qquad (3.2)$$

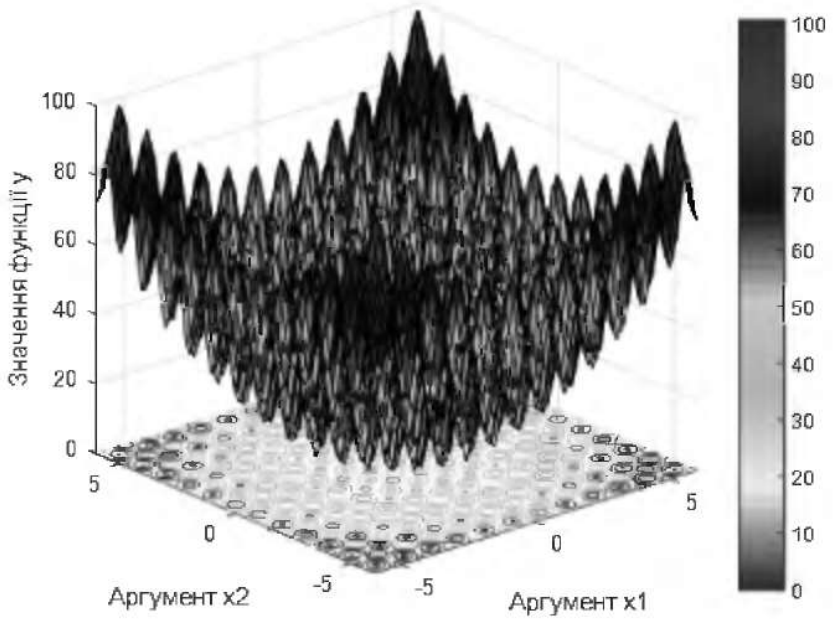


Рисунок 3.3 — Функція Растрігіна

Результати роботи програми:

— $n$ = 20, кількість знаходжень глобального мінімуму 38%, кількість обчислень цільової функції не більша 5100, максимальне значення функції 21,8572;

— $n$ = 30, кількість знаходжень глобального мінімуму 38%, кількість обчислень цільової функції не більша 11000, максимальне значення функції 18,9887;

— $n$ = 50, кількість знаходжень глобального мінімуму 36%, кількість обчислень цільової функції не більша 11000, максимальне значення функції 51,679.

3. Функція Розенброка. Глобальний оптимум знаходиться всередині параболічної форми поверхні, на повздовжній, досить вузькій та плоскій ”долині”. Знайти її нескладно, однак наблизитися до глобального оптимуму важко. Має один мінімум $F(x_i^*) = F(1,\ldots,1) = 0$ (рисунок 3.4).

$$F(x_1,\ldots,x_n) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2),\ x_i \in [-2;2]. \qquad (3.3)$$

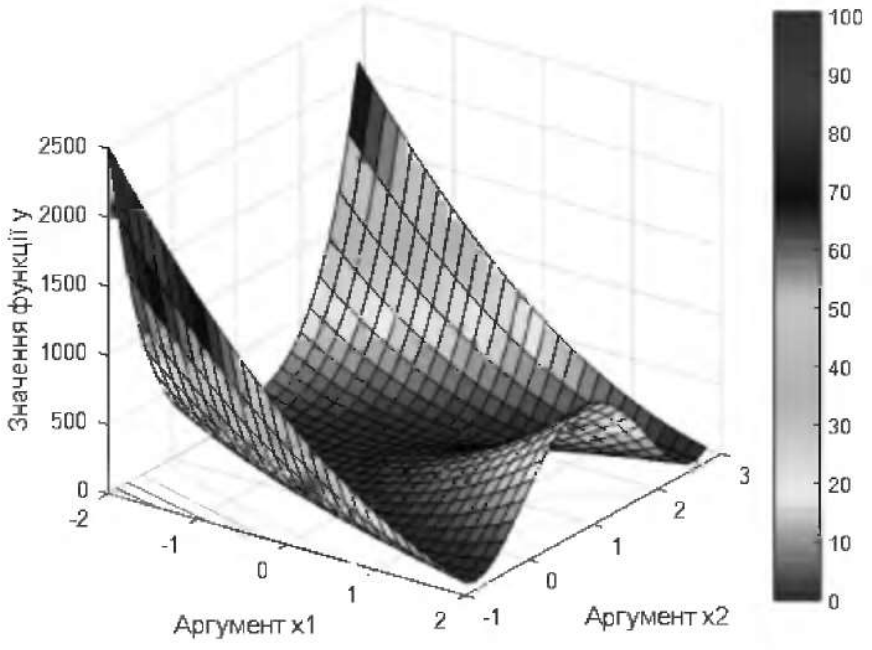


Рисунок 3.4 — Функція Розенброка

Результати роботи програми:

— $n$ = 2, кількість знаходжень глобального мінімуму 95%, кількість обчислень цільової функції не більша 1200, максимальне значення функції 0,00165226;

— $n$ = 4, кількість знаходжень глобального мінімуму 89%, кількість обчислень цільової функції не більша 1500, максимальне значення функції 0,00494682;

— $n$ = 10, кількість знаходжень глобального мінімуму 26%, кількість обчислень цільової функції не більша 15000, максимальне значення функції 0,0186537.

Результати роботи генетичного алгоритму показали його придатність для оптимізації функції багатьох змінних. Для функцій було знайдено глобальний мінімум зі збільшенням кількості змінних. Певні складності виникають з функцією (3.2), що пояснюється специфікою її рельєфу.

## 3.3 Висновки

У даному розділі представлено алгоритм роботи програмного забезпечення, а також наведено опис його параметрів. Тестування програмного забезпечення проведено на тестових функціях. Результати аналітичних обчислень підтвердили коректність роботи розробленого додатка. Експериментальні дослідження показують, що зі збільшенням кількості змінних від 10 до 50 для тестової функції 1 кількість знаходжень глобального мінімуму падає від 95% до 62%; зі збільшенням кількості змінних від 20 до 50 для тестової функції 2 — падає від 38% до 36%; зі збільшенням кількості змінних від 2 до 10 для тестової функції 3 — падає від 95% до 26%.

Результати експериментів показали, що острівну модель слід застосовувати тоді, коли немає спеціального алгоритму рішення заданої задачі.

# ВИСНОВКИ

У роботі проведено аналіз роботи генетичних алгоритмів, представлено теоретичну основу. Генетичний алгоритм — це евристичний поліноміальний алгоритм пошуку, який може застосовуватися для розв'язання різноманітних задач оптимізації і моделювання шляхом послідовного підбору, комбінування та варіювання шуканих параметрів.

У даній роботі розроблено програмне забезпечення, яке дозволяє дослідити ефективність роботи генетичного алгоритму під час оптимізації різних тестових функцій. Слід зазначити, що цільова функція повинна мати різноманітний рельєф, оскільки, якщо на її поверхні є великі плоскі ділянки (функція Розенброка), то генетичний алгоритм виявляється неефективним. Це пов'язано з тим, що більшість особин у популяції при розходженні в генотипі не відрізнятимуться фенотипом, тобто не дивлячись на те, що особини різні, вони мають майже однакову пристосованість, а значить алгоритм не має можливості вибрати краще рішення і напрямок подальшого розвитку. Тобто, коли весь простір пошуку абсолютно однаковий, за винятком лише однієї точки, яка і є оптимальним рішенням, то генетичний алгоритм просто зупиниться або буде виконувати випадковий пошук. Інша вимога до фітнес-функції полягає у мінімізації обчислювальних ресурсів при її оцінці, оскільки це впливає на швидкість алгоритму.

Отримані результати показали, що застосування генетичного алгоритму не може гарантувати знаходження найкращого рішення, але він з великою ймовірністю дає одне з оптимальних рішень, не потребуючи громіздких обчислень. Саме тому використання даного алгоритму необхідне лише в тих випадках, коли для вирішення заданої задачі немає відповідного спеціального алгоритму. Острівна модель допомагає підвищити точність пошуку за рахунок використання паралелізму.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фогель Ф. Генетика человека. Том 1 / Ф. Фогель, А. Мотульски. – М.: Мир, 1989. – 312 с.

2. Holland J.H. Adaptation in Natural and Artificial Systems. An Introductory Analysis with Application to Biology, Control, and Artificial Intelligence / J.H. Holland. – London: Bradford Book Edition, 1994. – 211 p.

3. Holland J.H. Adaptation in Natural and Artificial Systems / J.H. Holland. – MIT Press, 1992. – 225 p.

4. Goldberg D.E. Genetic Algorithms in Search, Optimization, and Machine Learning / D.E. Goldberg. – Boston: Addison-Wesley, 1989. – 405 p.

5. Эволюционные методы моделирования и оптимизации сложных систем / Е.С. Семенкин, М.Н. Жукова, В.Г. Жуков, И.А. Панфилов, В.В. Тынченко. – Красноярск: СФУ, 2007. – 310 с.

6. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М.: Горячая линия-Телеком, 2004. – 452 с.

7. Zitzler E. Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications / E. Zitzler. – Zurich: Swiss Federal Institute of Technology, 1999. – 122 p.

8. Курейчик В.М. Генетические алгоритмы / В.М. Курейчик. – Таганрог: ТРТУ, 1998. – 239 с.

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

10. Курейчик В.В. Анализ и обзор моделей эволюции / В.В. Курейчик, В.М. Курейчик, П.В. Сороколетов // Известия РАН ТиСУ. – 2007. – № 5. – С. 114-126.

11. Курейчик В.В. Концепция эволюционных вычислений, инспирированных природными системами / В.В. Курейчик, В.М. Курейчик, С.И. Родзин // Известия ЮФУ. – 2009. – № 4. – С. 16-24.

12. Кравець І.М. Оптимізація багатоекстремальних функцій за допомогою генетичного алгоритму / І.М. Кравець // Проблеми інформатизації та управління. – К.: Інститут комп'ютерних технологій Національного авіаційного університету, 2010. – № 2 (30). – С. 95-99.

13. Меняйлов Е.С. Обзор и анализ существующих модификаций генетических алгоритмов / Авиационно-космическая техника и технология // Е.С. Меняйлов. – 2015. – № 70. – С. 244-254.

14. Огляд метаевристичних методів розв'язання задачі пошуку мінімального гамільтонового циклу [Електронний ресурс]: матер. XLIX науково-технічної конференції факультету комп'ютерних систем і автоматики / Ю.Ю. Іванов, А. П. Саєнко, В. Є. Опанасюк, Ю. О. Маліцький, В. М. Саранчук. – Вінниця: ВНТУ, 12.03.2020. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2020/paper/view/8721/7856.

15. Панченко Т.В. Генетические алгоритмы: учебно-методическое пособие / Панченко Т.В. – Астрахань: Издательский дом «Астраханский университет», 2007. – 87 с.

16. Дмитриев С.В. Оптимизация многоэкстремальных функций с помощью гибридных генетических алгоритмов / С.В. Дмитриев, В.А. Тененев // Известия Института математики и информатики. – 2006. – С. 163-166.

17. Калініна І.В. Використання генетичних алгоритмів в задачах оптимізації / І.В. Калініна, О.І. Лісовиченко // Адаптивні системи автоматичного управління. – 2015. – № 1 (26). – С. 48-61.

18. Генетический алгоритм поиска экстремума [Электронный ресурс]. – Режим доступа: http://vuz.exponenta.ru.

19. Норенков И.П. Эвристики и их комбинации в генетических методах дискретной оптимизации / И.П. Норенков // Информационные технологии. – 1999. – №1. – С. 2-7.

20. Джонс М.Т. Программирование искусственного интеллекта в приложениях / М.Т. Джонс. – М.: ДМК Пресс, 2011. – 312 с.

21. Кононюк А.Ю. Нейронні мережі і генетичні алгоритми / А.Ю. Кононюк. – К., 2008. – 446 с.

22. Koza J.R. Genetic Programming: On the Programming of Computers by Means of Natural Selection / J.R. Koza. – London: The MIT Press, 1992. – 840 p.

23. Pohlheim H. Advanced Techniques for the Visualization of Evolutionary Algorithms / H. Pohlheim // Proceedings of 42 International Scientific Colloquium Ilmenau. – 1997. – V. 3. – P. 60-68.

24. Пантелеев А.В. Методы оптимизации в примерах и задачах: учеб. пособие / А.В. Пантелеев, Т.А. Летова. – М.: Высш. шк., 2005. – 544 с.

25. Іванов Ю.Ю. Методи штучного інтелекту та наука про дані: лекції, алгоритми та задачі / Ю.Ю. Іванов. – 2018. – 110 с.

26. Whitley D. A Genetic Algorithm Tutorial / D. Whitley // Statistics and Computing. – Springer Netherlands, 1994. – V. 4(2). – P. 65-85.

27. Алгоритмы, методы, исходники [Электронный ресурс] / Генетические алгоритмы. – Режим доступа: http://algolist.manual.ru/ai/ga/ga1.php.

28. Hiroyasu T. New Model of Parallel Genetic Algorithm in Multi-Objective Optimization Problems - Divided Range Multi-Objective Genetic Algorithm / T. Hiroyasu, M. Miki, S. Watanabe // The Proceedings of the 2000 Congress on Evolutionary Computation. – La Jolla (USA), 2000. – P. 333-340.

29. ИНТУИТ [Электронный ресурс] / Параллельные генетические алгоритмы. – Режим доступа: https://www.intuit.ru/studies/courses/14227/1284/lecture/24174?page=2.

30. Kotriakhova S.N. Parallel Evolutionary Algorithm in High-Dimensional Optimization Problem / S. N. Kotriakhova, M. M. Stepanova // Selected Papers of the 7th International Conference Distributed Computing and Grid-technologies in Science and Education. – M., 2016. – P. 328-332.

31. Ануфриев И.Е. MATLAB 7 / И.Е. Ануфриев, А.Б. Смирнов, Е.Н. Смирнова. – СПб.: БХВ-Петербург, 2005. – 1104 с.

32. Сергиенко А.Б. Тестовые функции для глобальной оптимизации / А.Б. Сергиенко. – Красноярск: Сибирский государственный аэрокосмический университет имени академика М.Ф. Решетнева, 2014. – 113 с.

# ДОДАТКИ

| | | | | | 08-02.БДР.009.00.000 ПД | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | Літ. | Маса | Масштаб |
| Змн. | Арк. | № докум. | Підпис | Дата | *Дослідження роботи острівного генетичного алгоритму. Структура еволюційної стратегії* | | | |
| Розроб. | | Опанасюк В.Є. | | | | Арк. | Аркушів 1 | |
| Перевір. | | Іванов Ю.Ю. | | | | | | |
| Т. Контр. | | Іванов Ю.Ю. | | | | | | |
| Реценз. | | | | | | *ВНТУ, гр. 1СІ-16б* | | |
| Н. Контр. | | Іванов Ю.Ю. | | | | | | |
| Затверд. | | Квєтний Р.Н. | | | | | | |

Додаток А

(обов’язковий)

Структура еволюційної стратегії


Рисунок А.1 — Структура еволюційної стратегії

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | 08-02.БДР.009.00.000 СП | | |
| | | | | | | Літ. | Маса | Масштаб |
| Змн. | Арк. | № докум. | Підпис | Дата | *Дослідження роботи острівного генетичного алгоритму. Схема програми* | | | |
| Розроб. | | Опанасюк В.Є. | | | | Арк. | Аркушів 1 | |
| Перевір. | | Іванов Ю.Ю. | | | | | | |
| Т. Контр. | | Іванов Ю.Ю. | | | | | | |
| Реценз. | | | | | | ВНТУ, гр. 1СІ-16б | | |
| Н. Контр. | | Іванов Ю.Ю. | | | | | | |
| Затверд. | | Квєтний Р.Н. | | | | | | |

Додаток Б

(обов'язковий)

Схема програми


Рисунок Б.1 — Схема програми (1 острів)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | 08-02.БДР.009.00.000 ПД | | |
| | | | | | | Літ. | Маса | Масштаб |
| Змн. | Арк. | № докум. | Підпис | Дата | *Дослідження роботи острівного генетичного алгоритму. Інтерфейс програмного забезпечення* | | | |
| Розроб. | | Опанасюк В.Є. | | | | Арк. | Аркушів 1 | |
| Перевір. | | Іванов Ю.Ю. | | | | | | |
| Т. Контр. | | Іванов Ю.Ю. | | | | | | |
| Реценз. | | | | | | ВНТУ, гр. 1СІ-16б | | |
| Н. Контр. | | Іванов Ю.Ю. | | | | | | |
| Затверд. | | Квєтний Р.Н. | | | | | | |

Додаток В

(обов'язковий)

Інтерфейс програмного забезпечення

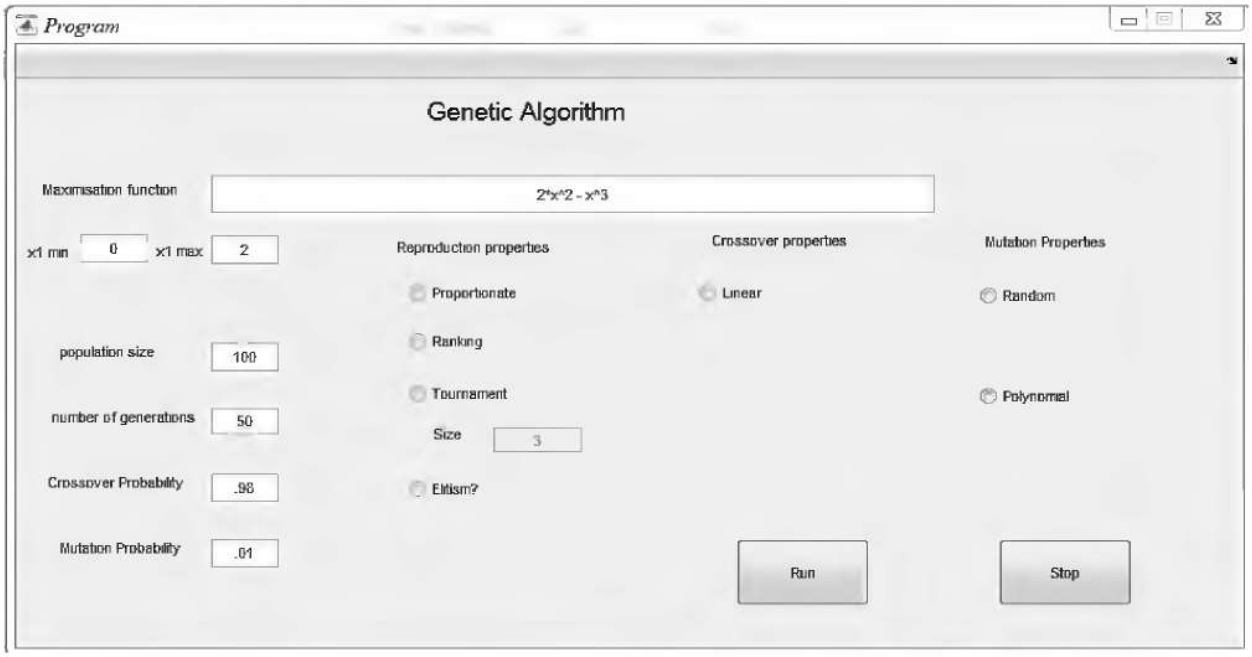


Рисунок В.1 — Графічний інтерфейс користувача

# Додаток Г

(обов'язковий)

## Лістинг програмного забезпечення

```
%-------------------------------------------------------------------------------------------------------------------------
Xmin(1) = -2; Xmax(1) = 2;
Xmin(2) = -2; Xmax(2) = 2;
Xmin(3) = -2; Xmax(3) = 2;
Xmin(4) = -2; Xmax(4) = 2;
points = 20;
vars = 4;

if vars == 2
   x1 = Xmin(1):0.01:Xmax(1);
   x2 = Xmin(2):0.01:Xmax(2);
   n1 = length(x1);
   n2 = length(x2);
   for i=1:1:n1
      for j=1:1:n2
         Y(i, j) = x1(i)^2 + x2(j)^2; % функція
      end
   end
   surfc(x1, x2, Y);
   colormap(hsv);
   colorbar;
   xlabel('Аргумент x1');
   ylabel('Аргумент x2');
   zlabel ('Значення функції y');
end

for i1=1:1:5
   switch i1
      case 1
         SF = @selectionremainder;
      case 2
         SF = @selectionuniform;
      case 3
         SF = @selectionstochunif;
      case 4
         SF = @selectionroulette;
      case 5
         SF = @selectiontournament;
   end

   PIR = [Xmin; Xmax];
   options = gaoptimset;
   options = gaoptimset (options, 'PopInitRange', PIR);
   options = gaoptimset (options, 'PopInitRange', []);
   options = gaoptimset (options, 'PopulationSize', 100);
   options = gaoptimset (options, 'EliteCount', 5);
```

```
    options = gaoptimset (options, 'CrossoverFraction', 1.0);
    options = gaoptimset (options, 'ParetoFraction', 0.50);
    options = gaoptimset (options, 'MigrationDirection', 'both');
    options = gaoptimset (options, 'MigrationInterval', 1);
    options = gaoptimset (options, 'MigrationFraction', 1);
    options = gaoptimset (options, 'Generations', 100);
    options = gaoptimset (options, 'TimeLimit', 400);
    options = gaoptimset (options, 'FitnessLimit', -Inf);
    options = gaoptimset (options, 'StallGenLimit', 50);
    options = gaoptimset (options, 'StallTimeLimit', 600);
    options = gaoptimset (options, 'TolFun', 1e-3);
    options = gaoptimset (options, 'TolCon', 1e-3);
    options = gaoptimset (options, 'InitialPopulation', []);
    options = gaoptimset (options, 'InitialScores', []);
    options = gaoptimset (options, 'InitialPenalty', []);
    options = gaoptimset (options, 'PenaltyFactor', []);
    options = gaoptimset (options, 'CreationFcn', @gacreationuniform);
    options = gaoptimset (options, 'FitnessScalingFcn', @fitscalingrank);
    options = gaoptimset (options, 'SelectionFcn', SF);
    options = gaoptimset (options, 'CrossoverFcn', {@crossoverheuristic, 1.2});
    options = gaoptimset (options, 'MutationFcn', @mutationadaptfeasible);
    options = gaoptimset (options, 'DistanceMeasureFcn', []);
    options = gaoptimset (options, 'HybridFcn', []);
    options = gaoptimset (options, 'Display', 'iter');
    options = gaoptimset (options, 'PlotFcns', {@gaplotbestf});
    options = gaoptimset (options, 'PlotInterval', 1);

    for k=1:1:points
        [X(k,:), FVal(k)]= ga(@optim, nvars, [], [], [], [], Xmin, Xmax, [], options);
    end

    F(i1) = mean(FVal);

    for j=1:1:vars
        X_(i1, j) = mean(X(:, j));
    end

    X_(i1,:);
end

plot (1:1:i1, F);
grid on;
%-----------------------------------------------------------------------------------------------------------------------

%-----------------------------------------------------------------------------------------------------------------------
function y = optim(x)
    result = 2*x(1)^2 + 3*x(2)^2 - x(3)^2 + x(4)^5;
end
%-----------------------------------------------------------------------------------------------------------------------
```