Прізвище: Тищенко

Ім’я: Данило

Група: КНМ-14

Дата прийняття роботи

у системі Git:

Дисципліна: Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні

Перевірив: Кривий Р.З.

**Звіт до лабораторної роботи № 4**

**«Реалізація генетичного алгоритму**

**на мові програмування Python»**

**МЕТА РОБОТИ**

Ознайомитися з основними теоретичними відомостями, вивчити еволюційні оператори схрещування та мутації, що використовуються при знаходженні значень цільової функції.

**КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ**

У загальному розумінні генетичні алгоритми (genetic algorithms) – це алгоритми, що використовують механізмами еволюції живої природи – природний відбір і генетичне наслідування. Генетичні алгоритми сьогодні застосовуються в різних галузях. Зокрема їх успішно використовують для розв’язування ряду важливих задач в економіці, бізнесі, техніці. З їх допомогою були розроблені промислові проектні рішення, що сприяли значній економії коштів і ресурсів. Фінансові компанії широко використовують ці засоби для прогнозування розвитку фінансових ринків для управління пакетами цінних паперів.

До основних характеристик ГА належать: розмір популяції (population size), оператор селекції (selection), оператор кросовера (crossover) і правила його використання, оператор мутації (mutation) і його параметри, оператор редукції (reduction), правило (критерій) зупинки процесу виконання генетичного алгоритму (stopping criteria). Оператори селекції, кросовера, мутації і редукції ще називають генетичними операторами.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

Розробити, за допомогою мови програмування Python, програмне забезпечення для знаходження значень цільової функції.

Функція:

*Де: a = 12, b = -8, c = -40, d = 3*

**Фрагмент програмного коду:**

**class XGene(FloatGene):**

**# genes get randomly generated within this range**

**randMin = RANDMIN**

**randMax = RANDMAX**

**# probability of mutation**

**mutProb = 0.2**

**# degree of mutation**

**mutAmt = 0.8**

**def func(x):**

**return a + b \* x + c \* x \*\* 2 + d \* x \*\* 3**

**class QuadraticSolver(Organism):**

**genome = {'max': XGene, 'min': XGene}**

**def fitness(self):**

**'''**

**try to minimize fitmess function**

**:return:**

**'''**

**x\_max = self['max']**

**x\_min = self['min']**

**max\_y = func(x\_max) # punish for incorrect first root**

**min\_y = func(x\_min) # punish for incorrect second root**

**return min\_y - max\_y**

**def \_\_repr\_\_(self):**

**return "<fitness=%f x1=%s x2=%s>" % (**

**self.fitness(), self['x1'], self['x2'])**

**class QPopulation(Population):**

**species = QuadraticSolver**

**initPopulation = 2**

**# cull to this many children**

**childCull = 10**

**# number of children**

**childCount = 25**

**# create a new population, with randomly created members**

**pop = QPopulation()**

**# now a func to run the population**

**def main():**

**x\_max, x\_min = 0, 0**

**xs\_min = []**

**xs\_max = []**

**try:**

**generations = 100**

**for i in range(generations):**

**# execute a generation**

**pop.gen()**

**best = pop.organisms[0]**

**print("fitness=%f x1=%f x2=%f" % (best.get\_fitness(), best['max'], best['min']))**

**xs\_min.append(best['min'])**

**xs\_max.append(best['max'])**

**x\_max = best['max']**

**x\_min = best['min']**

**except KeyboardInterrupt:**

**pass**

**print("Executed", generations, "generations")**

**Результат виконання лабораторного завдання.**

1. Початкові умови:

Кількість ітерацій: 50

Імовірність мутації: 0.3

Вплив мутації: 0.5

Кількість нащадків: 50

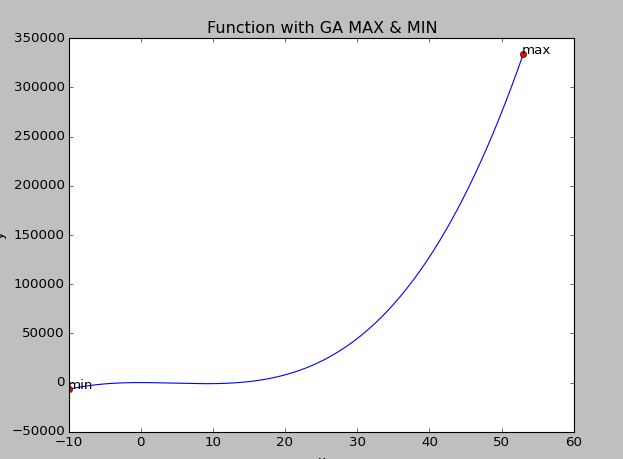


Рис.1 Цільова функція і знайдені мінімум та максимум за

початкових параметрів №1

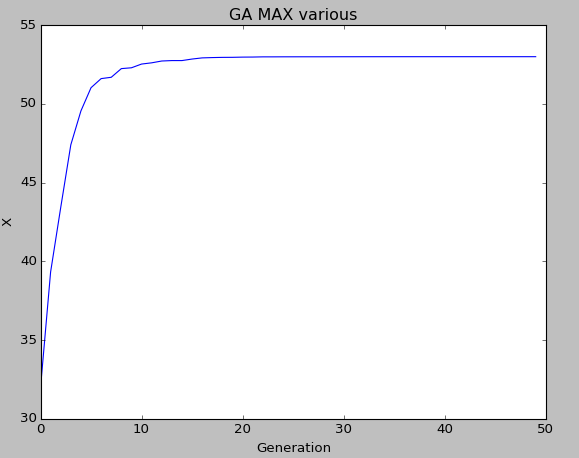


Рис.2 Варіація максимуму від номера генерації за

початкових параметрів №1

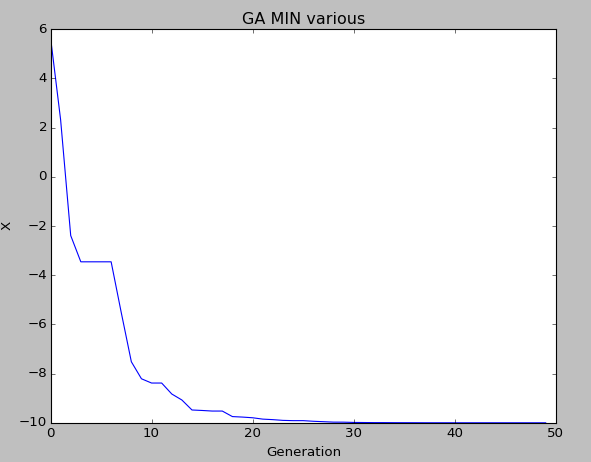


Рис.3 Варіація мінімуму від номера генерації за

початкових параметрів №1

1. Початкові умови:

Кількість ітерацій: 50

Імовірність мутації: 0.6

Вплив мутації: 0.2

Кількість нащадків: 75

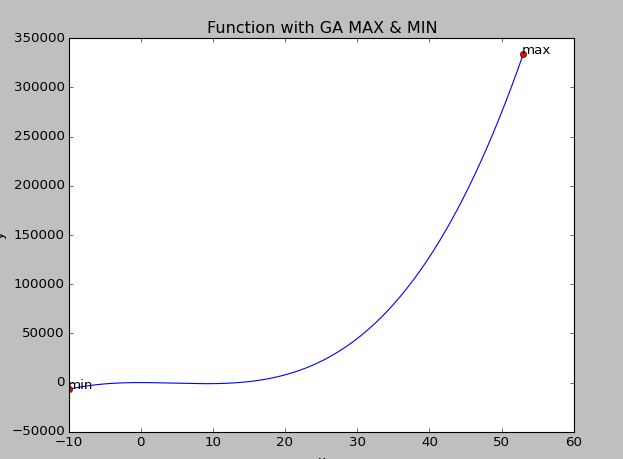


Рис.1 Цільова функція і знайдені мінімум та максимум за

початкових параметрів №2

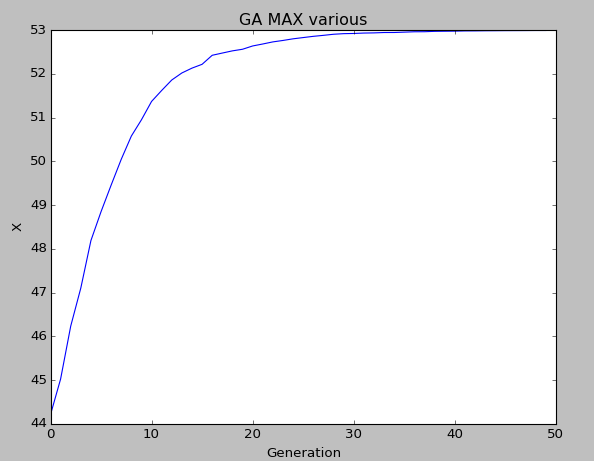


Рис.2 Варіація максимуму від номера генерації за

початкових параметрів №2

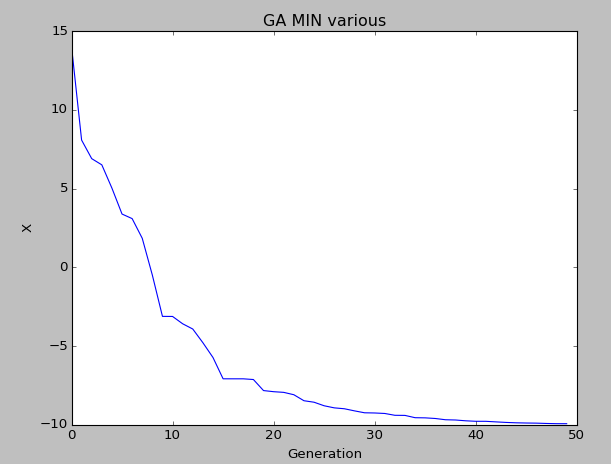


Рис.3 Варіація мінімуму від номера генерації за

початкових параметрів №2

1. Початкові умови:

Кількість ітерацій: 50

Імовірність мутації: 0.5

Вплив мутації: 0.5

Кількість нащадків: 10

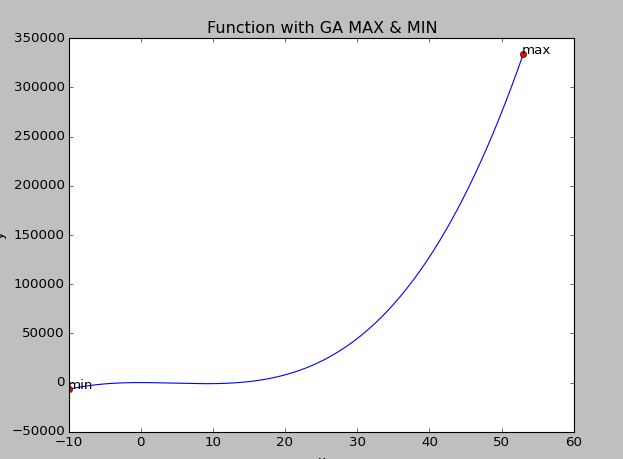


Рис.1 Цільова функція і знайдені мінімум та максимум за

початкових параметрів №3

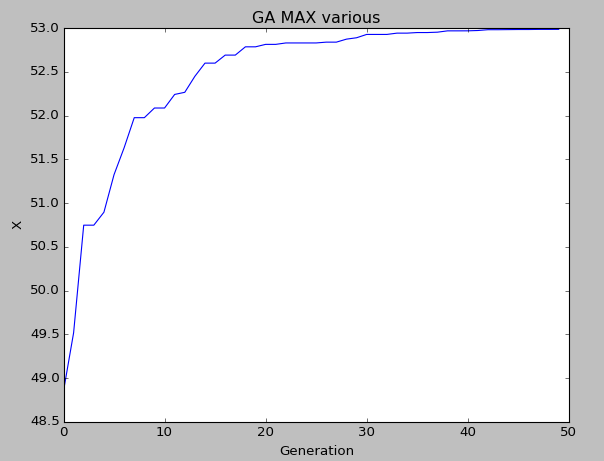


Рис.2 Варіація максимуму від номера генерації за

початкових параметрів №3

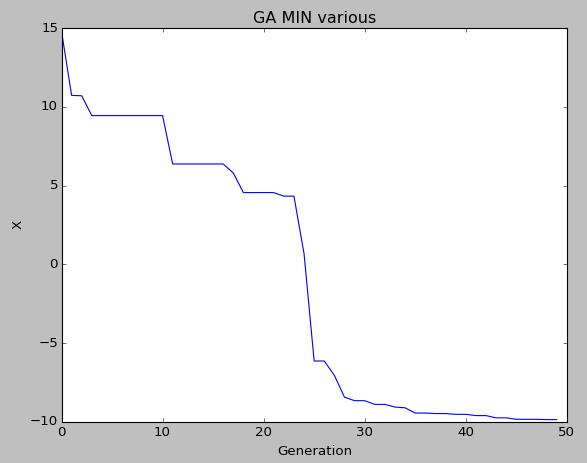


Рис.3 Варіація мінімуму від номера генерації за

початкових параметрів №3

1. Початкові умови:

Кількість ітерацій: 25

Імовірність мутації: 0.3

Вплив мутації: 0.1

Кількість нащадків: 50

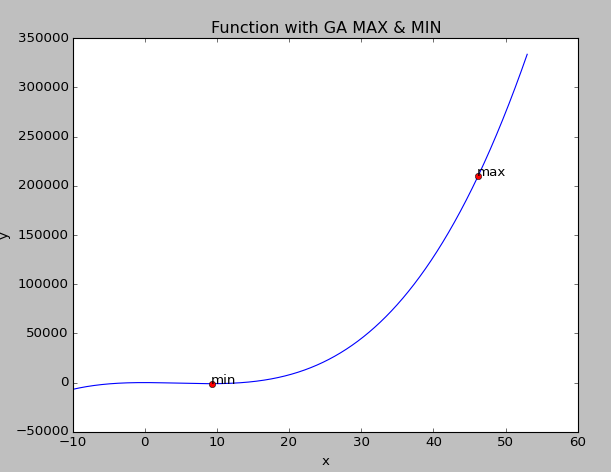


Рис.1 Цільова функція і знайдені мінімум та максимум за

початкових параметрів №4

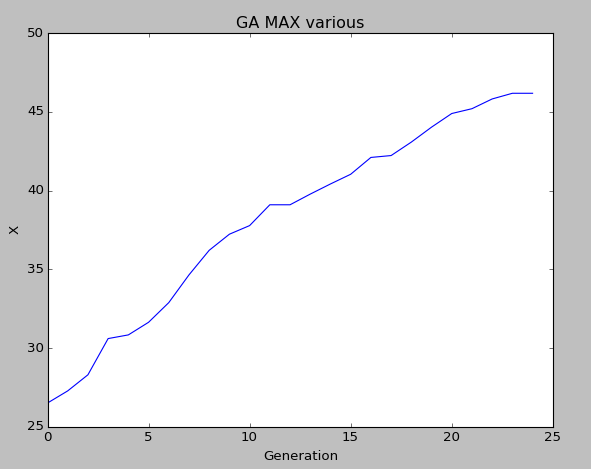


Рис.2 Варіація максимуму від номера генерації за

початкових параметрів №4

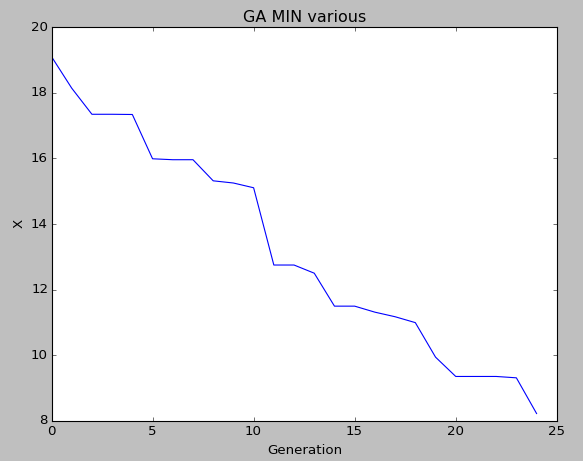


Рис.3 Варіація мінімуму від номера генерації за

початкових параметрів №4

1. Початкові умови:

Кількість ітерацій: 100

Імовірність мутації: 0.2

Вплив мутації: 0.8

Кількість нащадків: 25

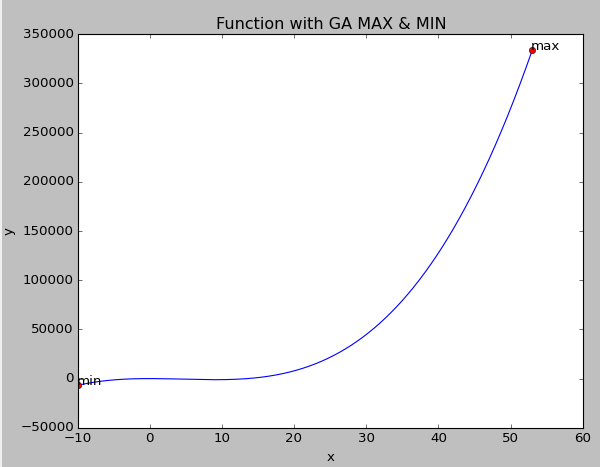


Рис.1 Цільова функція і знайдені мінімум та максимум за

початкових параметрів №5

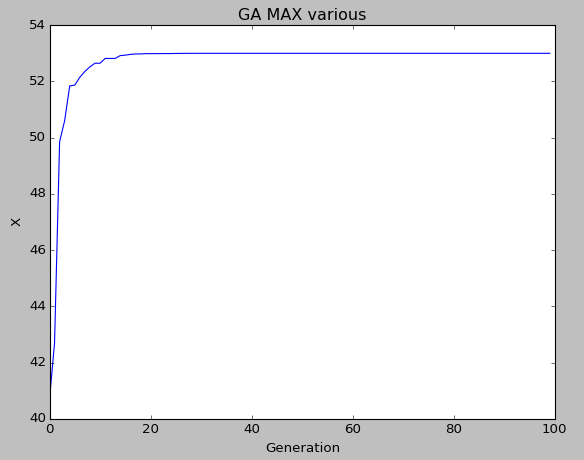


Рис.2 Варіація максимуму від номера генерації за

початкових параметрів №5

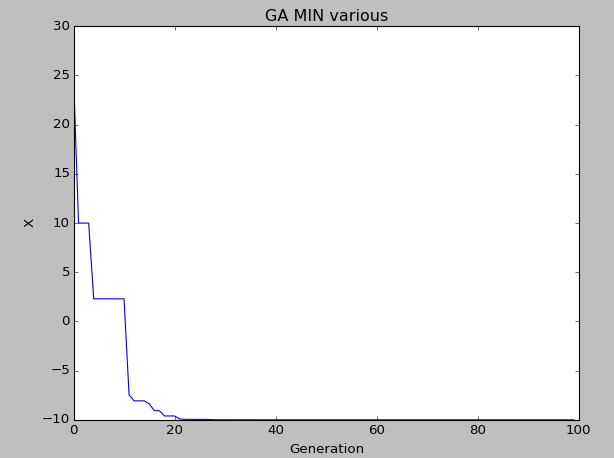


Рис.3 Варіація мінімуму від номера генерації за

початкових параметрів №5

**Висновки.**

Виконуючи лабораторну роботу я ознайомився з основними теоретичними відомостями, вивчив еволюційні оператори схрещування та мутації.

Реалізував програму, використовуючи мову програмування Python, для визначення значень цільової функції.