МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №3.3
з дисципліни «Інтелектуальні вбудовані системи»

на тему «Дослідження генетичного алгоритму »

Виконав:
студент гр. ІП-84
Дмитренко Олександр

Перевірив:
Регіда П.Г.

Київ 2021

Основні теоретичні відомості

Генетичні алгоритми служать, головним чином, для пошуку рішень в багатовимірних просторах пошуку.

Можна виділити наступні етапи генетичного алгоритму:

- (Початок циклу)
- Розмноження (схрещування)
- Мутація
- Обчислити значення цільової функції для всіх особин
- Формування нового покоління (селекція)
- Якщо виконуються умови зупинки, то (кінець циклу), інакше (початок циклу).

Розглянемо приклад реалізації алгоритму для знаходження цілих коренів діофантового рівняння $a+b+2c=15$.

Згенеруємо початкову популяцію випадковим чином, але з дотриманням умови – усі згенеровані значення знаходяться у проміжку від одиниці до $y/2$, тобто на відрізку [1;8] (узагалі, границі випадкового генерування можна вибирати на свій розсуд):

(1,1,5); (2,3,1); (3,4,1); (3,6,4)

Отриманий генотип оцінюється за допомогою функції пристосованості (fitness function). Згенеровані значення підставляються у рівняння, після чого обраховується різниця отриманої правої частини з початковим y. Після цього рахується ймовірність вибору генотипу для ставання батьком – зворотня дельта ділиться на сумму сумарних дельт усіх генотипів.

1+1+2·5=12	Δ=3	$\frac{\frac{1}{3}}{\frac{27}{24}} = 0{,}7$
2+3+2·1=7	Δ=8	$\frac{\frac{1}{8}}{\frac{27}{24}} = 0{,}11$
3+4+2·1=9	Δ=6	$\frac{\frac{1}{6}}{\frac{27}{24}} = 0{,}15$
3+6+2·4=17	Δ=2	$\frac{\frac{1}{2}}{\frac{27}{24}} = 0{,}44$

Наступний етап включає в себе схрещування генотипів по методу кросоверу – у якості дітей виступають генотипи, отримані змішуванням коренів – частина йде від одного з батьків, частина від іншого, наприклад:

$$\left.\begin{matrix}(3 \mid 6,4)\\(1 \mid 1,5)\end{matrix}\right\} \rightarrow \left\{\begin{matrix}(3,1,5)\\(1,6,4)\end{matrix}\right.$$

Лінія кросоверу може бути поставлена в будь-якому місці, кількість потомків також може вибиратися. Після отримання нових генотипів вони перевіряються функцією пристосованості та створюють власних потомків, тобто виконуються дії, описані вище.

Ітерації алгоритму відбуваються, поки один з генотипів не отримає Δ=0, тобто його значення будуть розв'язками рівняння.

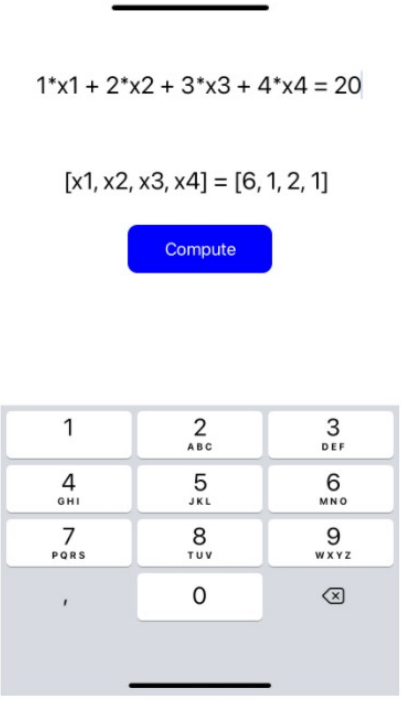
Завдання

Налаштувати генетичний алгоритм для знаходження цілих коренів діофантового рівняння ax1+bx2+cx3+dx4=y. Розробити відповідний мобільний додаток і вивести отримані значення. Провести аналіз витрат часу на розрахунки

Лістинг програми

```
const random = (min, max) => ~~(Math.random() * (max - min) + min)

class Chromosome {
    genes = [];
    fitness = Infinity;
    task = [];
    target = 0;

    calc = () =>
        this.genes.reduce((a, gene, i) => a + (gene * this.task[i]))

    constructor({ genes, task, target }) {
        Object.assign(this, { genes, task, target });
        this.calcFitness();
    }

    crossover(partner) {
        const child = this.clone();
        const fromParent = child.genes.length / 2;
        child.genes = [...child.genes.slice(0, fromParent), ...partner.genes.slice(child.genes.length - fromParent)];
        child.calcFitness();

        return child
    }

    calcFitness() {
        this.fitness = Math.abs(this.target - this.calc());
    }

    clone() {
        return Object.assign(Object.create(Object.getPrototypeOf(this)), this);
    }
}

class Genetic {
    population = []
    constructor(task, target) {
        const { length } = task
        this.population =
            Array.from(
                { length },
                () => new Chromosome({
                    genes: Array.from({ length }, () => random(1, target / 2)),
                    task: task,
                    target: target,
                })
            )
    }
    solve() {
        while (true) {
            const chromosome = this.crossover()
            if (chromosome)
                return chromosome.genes
        }
    }

    crossover() {
        const children = []
        for (let i = 0; i < this.population.length; i++) {
            const parents = this.population
                .map((chromosome) => ({ chromosome, probability: Math.random() * (chromosome.fitness * 1000) }))
                .sort((a, b) => a.probability - b.probability)

            const parent = parents[0].chromosome
            const partner = parents[1].chromosome
            const child = parent.crossover(partner)

            if (child.fitness === 0)
                return child

            children.push(child)
        }

        this.population = children
    }
}

export default Genetic;
```

Результат роботи програми



Висновки

Під час виконання лабораторної роботи були досліджені принципи реалізації генетичного алгоритму. Було досліджено особливості даного алгоритму з використанням засобів моделювання і сучасних програмних оболонок. Було налаштовано генетичний алгоритм для знаходження цілих коренів діафантового рівняння ax1+bx2+cx3+dx4 = y, розроблено відповідну програму та реалізовано користувацький інтерфейс з можливістю вводу даних.