Прізвище: Якубенко

Ім’я: Віталій

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 C#»**

**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 С#, програмне забезпечення для знаходження значень цільової функції.

**Фрагмент програмного коду:**

//initialize var

int cLength = Convert.ToInt32(chromoLength.Text);

EliteSelection<int> sel1 = new EliteSelection<int>(4);

RouletteSelection<int> sel2 = new RouletteSelection<int>();

GreedyCrossover cros = new GreedyCrossover(-1, -1, matrix);

GoldenMutation<int> mut = new GoldenMutation<int>(cLength);

pop = new Population<int>(mut, cros, sel1, sel2, x => 1 / cros.CalcFitness(x),

Convert.ToDouble(mProb.Text), Convert.ToDouble(cProb.Text));

maxF = new double[Convert.ToInt32(expCount.Text), Convert.ToInt32(iterCount.Text) + 1];

avgF = new double[Convert.ToInt32(expCount.Text), Convert.ToInt32(iterCount.Text) + 1];

string bestChromo = null;

//experiments

for (int i = 0; i < Convert.ToInt32(expCount.Text); ++i)

{

//initial chromosomes

Trace.WriteLine("experiment #" + (i + 1).ToString());

Trace.Indent();

ChromosomesFromArray();

maxF[i, 0] = Math.Round(1 / pop.GetMaxFitness());

avgF[i, 0] = 1 / pop.GetPopulationFitness();

Trace.WriteLine("initial best fitness = " + Math.Round((1 / pop.GetMaxFitness())).ToString());

Trace.WriteLine("initia;best fitness chromo = " + pop.GetMaxChromo());

// iterations

for (int j = 0; j < Convert.ToInt32(iterCount.Text); ++j)

{

Trace.WriteLine("iteration #" + (j + 1).ToString());

Trace.Indent();

pop.Iteration();

maxF[i, j + 1] = Math.Round(1 / pop.GetMaxFitness());

avgF[i, j + 1] = 1 / pop.GetPopulationFitness();

Trace.WriteLine(" best fitness = " + Math.Round((1 / pop.GetMaxFitness())).ToString());

Trace.WriteLine("best fitness chromo = " + pop.GetMaxChromo());

Trace.Unindent();

}

if (Math.Round(1 / pop.GetMaxFitness()) < min)

{

min = Math.Round(1 / pop.GetMaxFitness());

bestChromo = pop.GetMaxChromo();

}

}

**Результат виконання лабораторного завдання.**

1. Результати експерименту після 5-ти ітерацій:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ітерація/ екперимент | 0 | 1 | 2 | 3 | 4 | 5 |
| 1 | 20 | 19 | 15 | 16 | 14 | 16 |

- Максимальне значення - 20

- Локальні максимуми – 19, 16

- Мінімальне значення - 14

- Локальний мінімум - 15

Рис.1 Результат цільової функції для

початкових параметрів

1. Результати другого експерименту після 5-ти ітерацій:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 20 | 21 | 15 | 20 | 18 | 15 |

Рис.2 Результат функції

1. Результати третього експерименту після 5-ти ітерацій:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
| 3 | 20 | 19 | 17 | 16 | 15 | 16 |

Рис.3 Результат функції

1. Результати четвертого експерименту після 5-ти ітерацій:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
| 4 | 20 | 17 | 16 | 16 | 14 | 16 |

Рис.4 Результат функції

1. Результати п'ятого експерименту після 5-ти ітерацій:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
| 5 | 20 | 21 | 18 | 15 | 17 | 15 |

Рис.5 Результат функції

**Висновки.**

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

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