МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Ижевский государственный технический университет

имени М.Т. Калашникова»

Институт «Информатика и вычислительная техника»

Кафедра «Автоматизированные системы обработки информации и управления»

Отчет

по лабораторной работе №2

по дисциплине «Системы искусственного интеллекта»

на тему «Генетический алгоритм».

Вариант 13.

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил:  студент гр. Б20-782-2 | | Р. А. Соловьев | |
|  | |  |
| Принял:  к.т.н., доцент, зав. каф. кафедры АСОИУ | М. Н. Мокроусов | | |

Ижевск 2023

1. Постановка задачи

1. Изучить простой генетический алгоритм, используя любые источники.

2. Подготовить тестовые данные.

3. Для реализации генетического алгоритма написать свой программный код или

адаптировать любой существующий программный код.

4. Провести эксперименты.

5. Оформить отчёт.

Для варианта 13:

Разработать программу, реализующую генетический алгоритм поиска максимального и минимального значений целевой функции f(x) = 62 - 1x - 86x2 + 2x3 в интервале x ϵ [-10, 53].

1. Поиск минимума и максимума

Вручную найдем минимум и максимум. Получим: y=-23515 при x = 29 и y = 64160 при x = 54

1. Результат работы

Воспользуемся программой для нахождения минимума и максимума с помощью генетичского алгоритма.

На рисунке 3.1 показан процесс нахождения минимума. Программ останавливается, если в течение 10-ти поколений программа выдает один и тот же ответ.

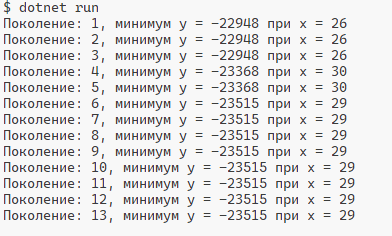


Рисунок 3.1 – Нахождение минимума

По тому же принципу найдем максимум. Процесс нахождения показан на рисунке 3.2

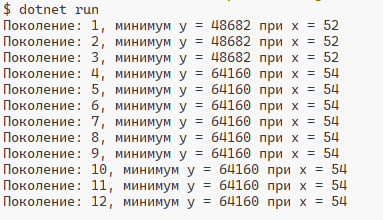


Рисунок 3.2 – Нахождение максимума

1. Исходный код программы
   1. Класс Population

На листинге 1 представлен класс, описывающий популяцию

namespace GeneticAlgorithm.Logic

{

    public class Population

    {

        private Random rnd;

        public List<Individual> individuals { get; private set; }

        public Population(Random rnd)

        {

            this.rnd = rnd;

        }

        public void SetRandomPopulation(List<Individual> individualsRange, int populationSize)

        {

            individuals = individualsRange.OrderBy(x => rnd.Next()).Take(populationSize).ToList();

        }

        public void SetNewPopulation(List<Individual> individualsRange)

        {

            individuals = individualsRange;

        }

    }

}

Листинг 1 – Класс Population.cs

* 1. Класс Individual

Опишем логику отдельной особи в классе Individual. Код класса представлен на листинге 2.

namespace GeneticAlgorithm.Logic

{

    public class Individual

    {

        public string BinaryGen { get; private set; }

        public int DecimalGen { get; private set; }

        public double Fitness { get; private set; }

        public int Size { get => BinaryGen.Length; }

        public Individual(int decimalGen = 0, string binaryGen = "")

        {

            if (binaryGen != "")

            {

                this.BinaryGen = binaryGen;

                this.DecimalGen = Convert.ToInt32(this.BinaryGen, 2);

            }

            else if (decimalGen != 0)

            {

                this.DecimalGen = decimalGen;

                this.BinaryGen = Convert.ToString(this.DecimalGen, 2);

            }

            else

            {

                this.DecimalGen = 0;

                this.BinaryGen = "0";

            }

            Fitness = FitnessFunction.Function(this);

        }

    }

}

Листинг 2 – Класс Individual.cs

* 1. Класс FitnessFunction

Опишем нашу функцию в классе. На листинге 3 представлен код класса FitnessFunction

namespace GeneticAlgorithm.Logic

{

    public class FitnessFunction

    {

        public static double Function(Individual individual)

        {

            int x = individual.DecimalGen;

            return 62 - x - 86 \* Math.Pow(x, 2) + 2 \* Math.Pow(x, 3);

        }

    }

}

Листинг 3 – Класс FitnessFunction.сs

* 1. Enum для выбора поиска

Для того, чтобы искать либо минимум функции, либо максимум, заведем перечисление, листинг 4

namespace GeneticAlgorithm.Logic

{

    public enum OptimizationWay

    {

        Min,

        Max

    }

}

Листинг 4 – Enum OptimizationWay

* 1. Основная логика алгоритма

Опишем основную логику программы, листинг 5

using System.Security.Cryptography.X509Certificates;

using System.Text;

namespace GeneticAlgorithm.Logic

{

    public class Algorithm

    {

        private double \_mutationPropability = 0.3;

        private int \_populationSize = 10;

        private int \_crossCount = 4;

        private int \_mutationCount = 1;

        // Если значение функции повторяется уже 4 раза, значит алгоритм пора останавилвть.

        private int answers = 10;

        private Individual Crossingover(Individual ind1, Individual ind2)

        {

            Random rnd = new Random();

            int minGenSize = Math.Min(ind1.Size, ind2.Size);

            int crossingoverDot = rnd.Next(1, minGenSize);

            string newGenes = ind1.BinaryGen.Substring(0, crossingoverDot) + ind2.BinaryGen.Substring(crossingoverDot);

            int decGen = Convert.ToInt32(newGenes, 2);

            // Проверка: Входит ли ребенок в границы

            if (decGen >= -10 && decGen <= 54)

                return new Individual(binaryGen: newGenes);

            else

                return ind1;

        }

        private Individual Mutate(Individual individual)

        {

            Random rnd = new Random();

            if (rnd.NextDouble() <= \_mutationPropability)

            {

                StringBuilder sb = new StringBuilder(individual.BinaryGen);

                int mutationIndex = rnd.Next(0, individual.Size);

                if (sb[mutationIndex] == '0')

                {

                    sb[mutationIndex] = '1';

                }

                else

                {

                    sb[mutationIndex] = '0';

                }

                string newGene = sb.ToString();

                int decGen = Convert.ToInt32(newGene, 2);

                // Проверка: Входит ли ребенок в границы

                if (decGen >= -10 && decGen <= 54)

                    return new Individual(binaryGen: newGene);

                else

                    return individual;

            }

            return individual;

        }

        private void SelectIndividuals(Population population, OptimizationWay way)

        {

            List<Individual> sortedPopulation;

            if (way == OptimizationWay.Max)

            {

                sortedPopulation = population.individuals.OrderByDescending(i => i.Fitness).Take(4).ToList();

            }

            else

            {

                sortedPopulation = population.individuals.OrderBy(i => i.Fitness).Take(4).ToList();

            }

            population.SetNewPopulation(sortedPopulation);

        }

        public void Run(List<Individual> allIndividuals)

        {

            Random rnd = new Random();

            Population population = new Population(rnd);

            population.SetRandomPopulation(allIndividuals, \_populationSize);

            int generation = 0;

            int dublicateAnswers = 0;

            double prevAnswer = 0;

            while (dublicateAnswers != answers)

            {

                // Делаем кроссинговеры

                for (int i = 0; i < \_crossCount; i++)

                {

                    var ind1 = population.individuals[rnd.Next(0, population.individuals.Count)];

                    var ind2 = population.individuals[rnd.Next(0, population.individuals.Count)];

                    var child = Crossingover(ind1, ind2);

                    // Пробуем мутировать и в итоге детей добавляем в популяцию

                    child = Mutate(child);

                    population.individuals.Add(child);

                }

                // Отбираем тех, у кого y - самый маленький, тк ищем минимум

                SelectIndividuals(population, OptimizationWay.Max);

                generation++;

                Console.WriteLine($"Поколение: {generation}, минимум y = {population.individuals.First().Fitness} при x = {population.individuals.First().DecimalGen}");

                if (prevAnswer == population.individuals.First().Fitness)

                    dublicateAnswers++;

                prevAnswer = population.individuals.First().Fitness;

            }

        }

    }

}

Листинг 5 – Алгоритм программы