

Министерство транспорта Российской Федерации
Федеральное агентство железнодорожного транспорта
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Дальневосточный государственный университет путей сообщения»
Кафедра «Вычислительная техника и компьютерная графика»

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ
Лабораторная работа №5
ЛР 09.04.01.МРО.08.05.МО921ИВС

Выполнил студент гр. МО921ИВС	_____	А.Ю. Панченко
Проверил доцент, к.ф.-м.н.	_____	Ю.В. Пономарчук

Цель работы: изучить основные особенности работы генетических алгоритмов, описать функционирование одной эпохи генетического алгоритма с заданными параметрами (фитнесс-функция, ранговый отбор, одноточечный кроссовер и одноточечная мутация).

1 УСЛОВИЕ ЗАДАЧИ

Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит особи, деленная на количество особей в популяции; метод отбора – ранговый с использованием принципа элитизма; оператор скрещивания – одноточечный кроссовер; оператор мутации – одноточечная мутация.

2 ОПИСАНИЕ ПРОЦЕССА РЕШЕНИЯ

Для использования генетического алгоритма необходимо:

- 1 Определить набор признаков, характеризующие решения задачи оптимизации или моделирования. Определить фенотип, закодировать признаки (можно использовать код Грея).

- 2 Использовать последовательность шагов генетического алгоритма с соответствующими операторами.

3 ХОД РЕШЕНИЯ

1) Фенотип (заданы случайные десятичные значения)

Признак	Двоичное значение	Десятичное значение	Код Грея
Признак 1	0010	2	0011
Признак 2	0101	5	0111
Признак 3	0110	6	0101
Признак 4	1000	8	1100
Признак 5	1011	11	1110

2)

1 шаг: Формирование начальной популяции

Используя 5 признаков (по 2 признака в особи), случайным образом сгенерирована популяция из 10 особей (каждая особь – 8 бит):

№ особи	Признаки	Хромосома
1	1+2	00100101
2	2+5	01011011
3	3+4	01101000
4	4+1	10000010
5	5+3	10110110
6	1+5	00101011
7	2+4	01011000
8	3+2	01100101
9	4+5	10001011
10	5+1	10110010

2 шаг: Оценка особей популяции (использование фитнес-функции)

Вычисляем сумму бит для каждой особи.

Размер популяции равен 10.

Фитнес-функция вычисляется по формуле:

$$fitness(i) = \frac{\sum bit(i)}{10}$$

Результат вычислений представлен на рисунке 1.

Fitness table:

Individual	Chromosome	Bit sum	Fitness
1	00100101	3	0,300
2	01011011	5	0,500
3	01101000	3	0,300
4	10000010	2	0,200
5	10110110	5	0,500
6	00101011	4	0,400
7	01011000	3	0,300
8	01100101	4	0,400
9	10001011	4	0,400
10	10110010	4	0,400

Рисунок 1 – Подсчет суммы битов

3 шаг: Отбор (ранговый отбор)

Особи сортируются по убыванию фитнес-функции с присвоением рангов (рисунок 2).

Rank	Original #	Chromosome	Fitness
10	2	01011011	0,500
9	5	10110110	0,500
8	6	00101011	0,400
7	8	01100101	0,400
6	9	10001011	0,400
5	10	10110010	0,400
4	1	00100101	0,300
3	3	01101000	0,300
2	7	01011000	0,300
1	4	10000010	0,200

Рисунок 2 – Присвоение рангов

При выборе пар вероятность попадания в пару рассчитывается следующим образом:

$$P_{in} = \frac{R_i}{\sum R}$$

В листинге 1 представлен код программы для составления пар.

Листинг 1 – Составление пар

```
public List<(int, int)>
CreatePairsByRankProbability(Dictionary<int, int> ranks)
{
    int totalRank = ranks.Values.Sum();
    int numPairs = (_config.PopulationSize - 2) / 2; // 2 элиты,
    остальные для пар

    List<int> SelectParents()
    {
        var cumulativeRanks = new List<(int index, int
cumulative)>();
        int cumulativeSum = 0;

        foreach (var rankPair in ranks)
        {
            cumulativeSum += rankPair.Value;
            cumulativeRanks.Add((rankPair.Key, cumulativeSum));
        }

        List<int> selected = new();

        while (selected.Count < 2)
        {
            int randomValue = _random.Next(totalRank);
            int chosenIndex = cumulativeRanks.First(pair =>
randomValue < pair.cumulative).index;

            if (!selected.Contains(chosenIndex))
                selected.Add(chosenIndex);
        }

        return selected;
    }

    var pairs = new List<(int, int)>();

    for (int i = 0; i < numPairs; i++)
    {
        var parents = SelectParents();
```

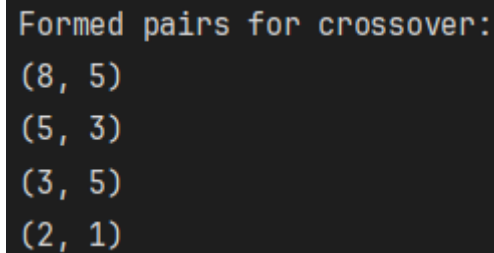
```

        pairs.Add((parents[0], parents[1]));
    }

    return pairs;
}

```

Пример составленных пар представлен на рисунке 3.



```

Formed pairs for crossover:
(8, 5)
(5, 3)
(3, 5)
(2, 1)

```

Рисунок 3 - Пары

При этом имеется две элитные особи (2 и 5), которые передаются без изменений.

Таким образом, в следующем поколении будет 8 наследников и 2 элитные особи, то есть также 10 особей.

4 шаг: Скрещивание (оператор одноточечного кроссовера)

Одноточечный кроссовер выбирается одна точка разрыва и родительские хромосомы обмениваются одной из получившихся частей.

Скрещивание производилось при помощи программного кода, представленного в листинге 2.

Листинг 2 – Программная реализация скрещивания (одноточечный кроссовер)

```

public List<List<int>> Crossover(List<List<int>> population,
List<(int, int)> pairs)
{
    var offspring = new List<List<int>>();

    foreach ((int i1, int i2) in pairs)
    {
        List<int> p1 = population[i1];
        List<int> p2 = population[i2];
        int breakPoint = _random.Next(minValue: 1,
_config.IndividualLength - 1);

        List<int> child1 = p1
            .Take(breakPoint)
            .Concat(p2.Skip(breakPoint))
            .ToList();
    }
}

```

```

        List<int> child2 = p2
            .Take(breakPoint)
            .Concat(p1.Skip(breakPoint))
            .ToList();

        offspring.Add(child1);
        offspring.Add(child2);
    }

    return offspring;
}

```

Результат скрещивания (рисунок 4)

```

Single point crossover results:
Pair 1: Individual 8 (01100101), Individual 5 (10110110)
    Descendant 1: 01100110
    Descendant 2: 10110101
Pair 2: Individual 5 (10110110), Individual 3 (01101000)
    Descendant 1: 10101000
    Descendant 2: 01110110
Pair 3: Individual 3 (01101000), Individual 5 (10110110)
    Descendant 1: 01101110
    Descendant 2: 10110000
Pair 4: Individual 2 (01011011), Individual 1 (00100101)
    Descendant 1: 01000101
    Descendant 2: 00111011

```

Рисунок 4 – Результат скрещивания

5 шаг: Мутация (одноточечная)

Для дальнейшей оптимизации и повышения разнообразия популяции на некоторых потомках применяется оператор одноточечной мутации: произвольный бит хромосомы с определенной вероятностью заменяется на противоположный.

Программный код мутации представлен в листинге 3.

Листинг 2 – Программная реализация мутации

```

public List<List<int>> MutateOffspring(List<List<int>>
offspring)
{
    return offspring
        .Select(
            child =>
            {
                if (_random.NextDouble() <
_config.MutationProbability)
                {
                    var mutated = new List<int>(child);
                    int index =
_random.Next(_config.IndividualLength);
                    mutated[index] = 1 - mutated[index];
                    return mutated;
                }

                return child;
            }
        )
        .ToList();
}

```

Результат мутации с использованием вероятности 50% представлен на рисунке 5.

```

Applying mutation:
Mutation 1: 01100110 => 01100111
Unchanged 2: 10110101
Mutation 3: 10101000 => 10101100
Unchanged 4: 01110110
Mutation 5: 01101110 => 01111110
Mutation 6: 10110000 => 10111000
Mutation 7: 01000101 => 11000101
Unchanged 8: 00111011

```

Рисунок 5 – Результат мутации

6 шаг: Формирование новой популяции

Новая популяция формируется на основе потомков после применения мутации (рисунок 6).


```
New population
Individual 1: 01011011
Individual 2: 10110110
Individual 3: 01100111
Individual 4: 10110101
Individual 5: 10101100
Individual 6: 01110110
Individual 7: 01111110
Individual 8: 10111000
Individual 9: 11000101
Individual 10: 00111011
```

Рисунок 6 – Новая популяция

7 шаг: Проверка разнообразия популяции

Анализ новой популяции показывает удовлетворительное разнообразие генотипов. Каждый потомок имеет уникальное распределение бит, что предотвращает преждевременную сходимость алгоритма.

Поскольку рассматривается лишь одна эпоха – выход из алгоритма.

Вывод: ходе работы была описана и реализована одна эпоха генетического алгоритма с заданными параметрами (фитнесс-функция, ранговый отбор с элитизмом, одноточечный кроссовер, одноточечная мутация) для оптимизации популяции, состоящей из десяти особей с пятью признаками.