



Разработка обучающего модуля для работы с генетическими алгоритмами

Выполнила: ст.гр. ИТ-51

Пискулина М. С.

Дипломный руководитель:

Катковская К. В.

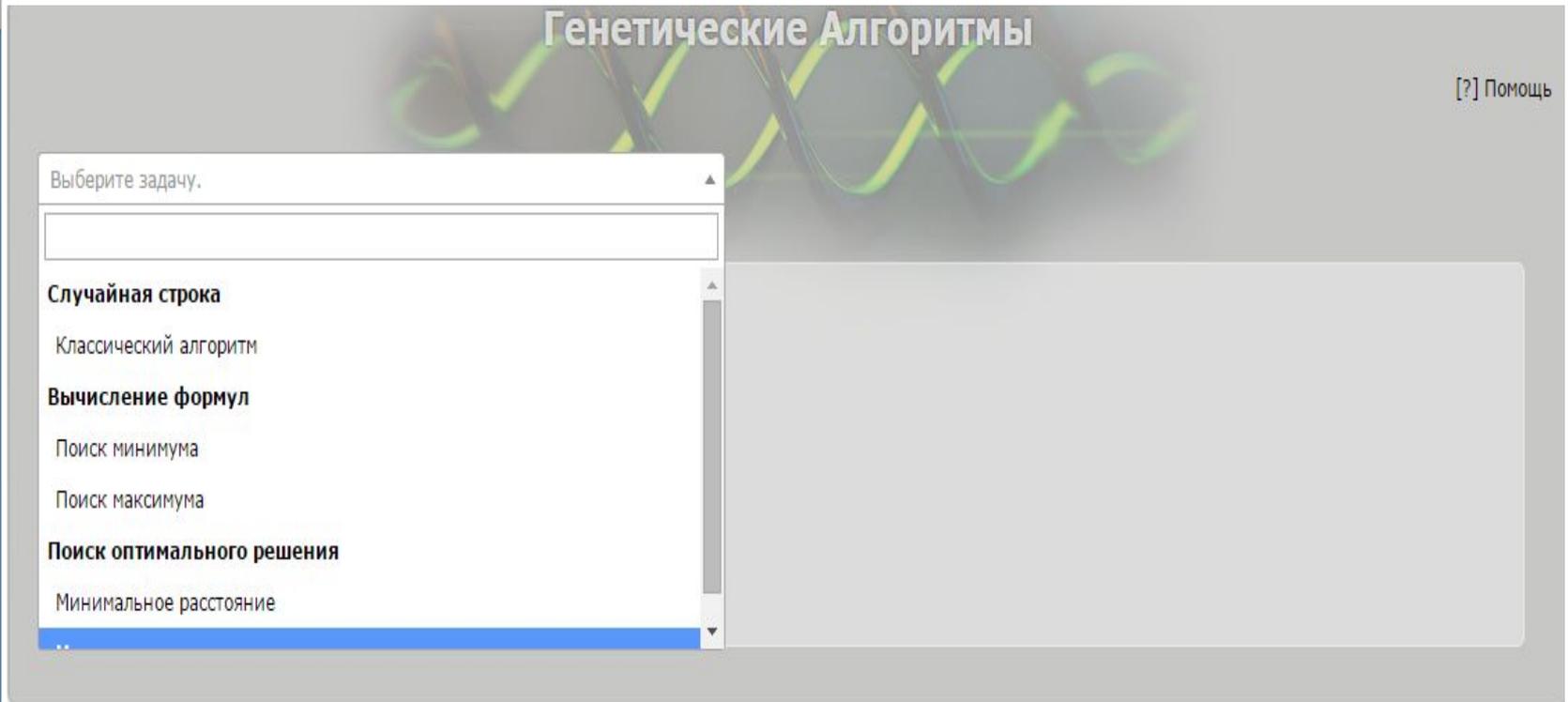
Задачи программного модуля

- 1 Возможность реализации различных видов генетических алгоритмов;
- 2 Возможность выбора тематики различных классов задач;
- 3 Наглядное интерактивное и динамическое представление этапов решения задач, в зависимости от выбранного алгоритма;
- 4 Наличие справки по использованию модуля.

	ИС «GenSearch»	Flex Tool	3Genetic	ECL
Язык программирования	Delphi	Matlab	Java	Java
Доступ	Платная	Платная	Платная	Бесплатная
Интерфейс	Текстовый	Текстовый	Графический	Графический
Задачи	Осуществление генетического поиска, обеспечение настроек и управление всеми его элементами.	Выбор ГА, выбор метода селекции и способа кодирования. Нахождение более чем одного оптимума.	Предназначена для решения задач различных областей при помощи ГА, а также как средство тестирования и сравнения различных ГА.	Поддерживает ряд технологий эволюционных вычислений, таких как, ГА, генетическое программирование, эволюционные стратегии, оптимизация большого числа частиц.
Возможность доступного освоения шагов алгоритма.	нет	нет	нет	нет
Возможность доступного обучения пользователя.	нет	нет	нет	нет

Общая схема модуля

Внешний вид программного модуля для решения задач



Нахождение минимального расстояния между городами

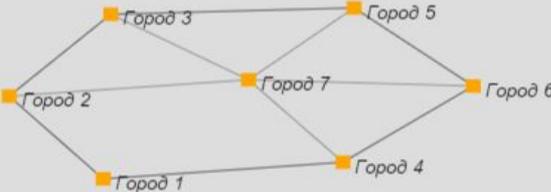
Генетические Алгоритмы

[?] Помощь

Минимальное расстояние

Откуда	Куда	Время в пути(минты)	Стоимость(руб)
Город 1	Город 2	40	50
Город 2	Город 3	120	200
Город 1	Город 4	60	70
Город 2	Город 1	20	15

Отправление: Прибытие: Максимальное поколение:



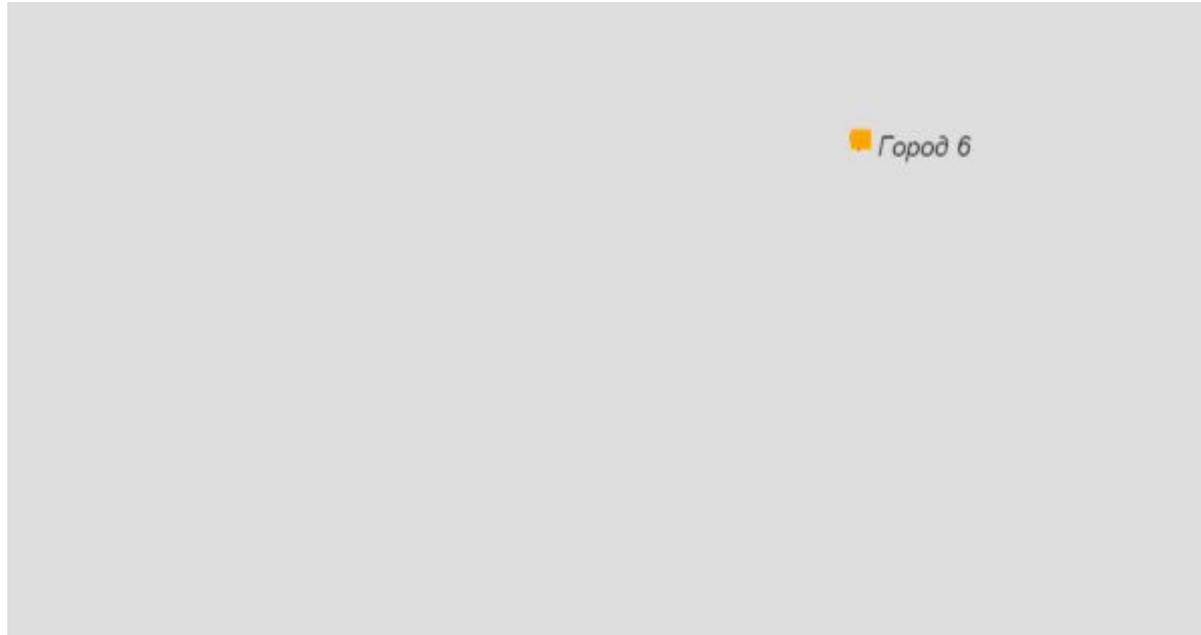
Наилучший вариант маршрута:
Город 1 > Город 2 > Город 3 > Город 5
Обойдется в сумму: **300 рублей.**
Займет времени: **220 минут.**

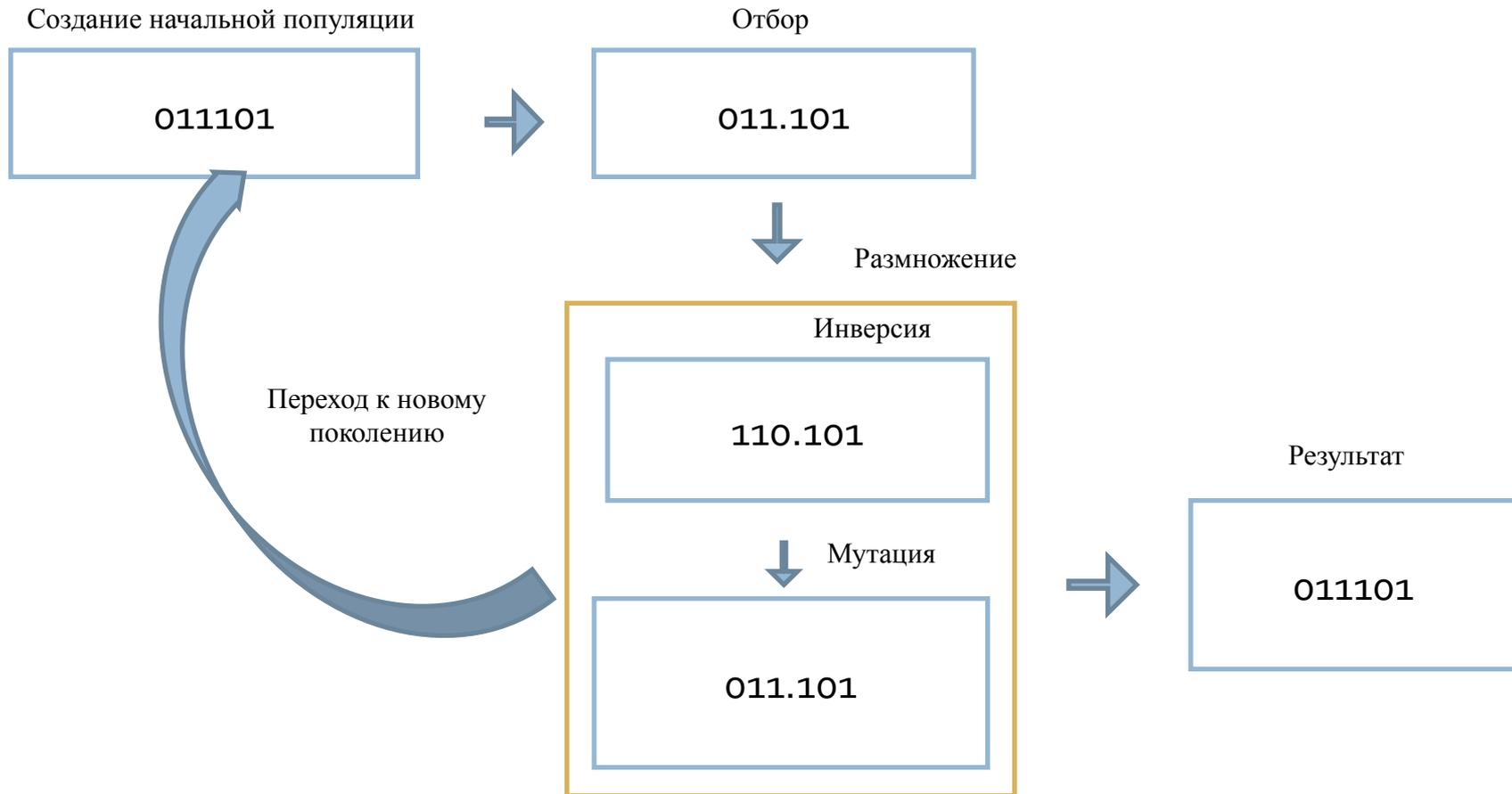
Прочие топ-5:

Маршрут:
Город 1 > Город 4 > Город 6 > Город 5
Обойдется в сумму: **240 рублей.**
Займет времени: **220 минут.**

Маршрут:

Пример динамичности программы





Создание начальной популяции

```

1 //Изначальный геном
2 initialGenome: function
  (fromCityName) {
3 //Если город найден в (переменной
  geoCities = база)
4 if(geoCities.indexOf(fromCityName)>-) {
5 // Пути
6 var wayIdx = [];
7 //Фильтр, берем только те точки,
  которые подходят
  по месторасположению, и каждую
8 $.grep(geoDATA, function(n) { return
  n.from == fromCityName;
9 }).map(function(n) {
10 //Запоминаем в wayIdx
11 wayIdx.push([$ (geoDATA).index(n)]);
12 });
13 //Записываем это как новый геном
14 this.genome = wayIdx;
15 return true;
16 } else {
17 return false;;}

```



Отбор

```

1 //Фитнесс функция
2 fitness: function () {
3 var fitness = [];
4 //Каждый элемент генома
5 $(this.genome).map(function(i,elem) {
6 //Запомним последний элемент генома
7 var lastElem = elem[elem.length-1];
8 // Если точка отправления последнего
9 элемента равна конечной
10 if(geoDATA[lastElem].to == geneticTSP.to) {
11 //Запишем
12 fitness.push(elem)}});
13 //Каждый новый найденный маршрут
14 var asa = $(fitness).map(function
  (i, elem) {
15 //Соединяем через запятую
16 return elem.join(',');
17 }).get();
18 var newFitness = [];
19 //Каждый найденный маршрут
20 asa.map(function(i) {
21 if(newFitness.indexOf(i)<0) {
22 newFitness.push(i);});
23 newFitness = $(newFitness).map(function
  (i, elem) {
24 return [elem.split(',')]);}).get();
25 this.genome = newFitness;}

```

Размножение

```

1  mutation: function () {
2  var newGenome = [];
3  //Каждый элемент генома
4  $(this.genome).map(function (i, elem) {
5  //Последний элемент генома
6  var lastElem = elem[elem.length-1];
7  //Если это конечный маршрут - больше не
строим его
8  if(geoDATA[lastElem].to == geneticTSP.to)
return;
9  //Ищем новые варианты путей
10 var newWays =
geneticTSP.getWaysByCheckpoint (geoDATA[lastElem]
.to);
11 //Все найденные пути
12 $(newWays).map(function (way) {
13 //Записываем в новый геном
14 newGenome.push(elem.concat (parseInt(this)));
15 });
16 });
17 //Следующее поколение
18 this.generation++;
19 //Записываем обновленные маршруты
20 this.genome = this.genome.concat (newGenome);
21 }

```

**Результат**

```

1  ...
2  $('<div class="frame output">').html (html);
3  //записываем вывод результата в
окно с классом .output
4  sys.renderer =
Renderer("#<div class="viewport">")
5  //начинаем рисовать в выбраной
области график
6  $.each(geoCities,
function(i,node) {
7  sys.addNode (node); //добавляем
вершину});
8  $.each(geoDATA, function(i,edge) {
9  sys.addEdge (sys.getNode (edge.from) ,
sys.getNode (edge.to));
10 });
11 ...

```

Реализация задачи «Случайная строка»

The screenshot shows a web-based interface for a genetic algorithm simulation. At the top, the title "Генетические Алгоритмы" is displayed in a light green font, with a background image of a DNA double helix. In the top right corner, there is a link "[?] Помощь". Below the title, there is a dropdown menu currently set to "Классический алгоритм". The interface is divided into two columns of input fields. The left column contains: "Ключ:" with the value "Hello Word!", "Плодovitость:" with the value "10", and "Максимальное поколение:" with the value "50". The right column contains: "Выживаемость:" with the value "500" and "Стартовая популяция:" with the value "1000". A large "Запуск" button is positioned below these fields. At the bottom, there is a scrollable text area showing the results of the simulation. It displays the top 5 strings from the 1st and 2nd generations. The 1st generation strings are: "=envqЦWЕАШЖШ", "=envqЦWЕАШЖШ", "=enviЦWЕАШЖШ", "OenvqЦWЕАШЖШ", and "УenvqЦWЕАШЖШ". The 2nd generation strings are: "=env&FWЕАШЖШ", "=envqЦWЕАШЖШ", "OenvqЦWЕАШЖШ", "УenvqЦWЕАШЖШ", and "деnvqЦWЕАШЖШ".

Генетические Алгоритмы

[?] Помощь

Классический алгоритм

Ключ: Hello Word!

Выживаемость: 500

Плодovitость: 10

Стартовая популяция: 1000

Максимальное поколение: 50

Запуск

1 поколение, 5 лучших:
=envqЦWЕАШЖШ
=envqЦWЕАШЖШ
=enviЦWЕАШЖШ
OenvqЦWЕАШЖШ
УenvqЦWЕАШЖШ

2 поколение, 5 лучших:
=env&FWЕАШЖШ
=envqЦWЕАШЖШ
OenvqЦWЕАШЖШ
УenvqЦWЕАШЖШ
деnvqЦWЕАШЖШ

Нахождение «Минимума функции»

Генетические Алгоритмы

Поиск минимума

Формула: $y(x) = A + B \cdot x + C \cdot x^2 + D \cdot x^3 + E \cdot x^4$

Коэффициент A:	Коэффициент B:	Коэффициент C:	Коэффициент D:	Коэффициент E:
<input type="text" value="5"/>	<input type="text" value="-24"/>	<input type="text" value="17"/>	<input type="text" value="-3.66666"/>	<input type="text" value="0.25"/>
Плодовитость:	Выживаемость:	Стартовая популяция:	Максимальное поколение:	<input type="button" value="Запуск"/>
<input type="text" value="10"/>	<input type="text" value="20"/>	<input type="text" value="100"/>	<input type="text" value="20"/>	

15 поколение, 5 лучших:
X: 0.9999986707271369, Y: -5.416660000013338
X: 0.9999986693949083, Y: -5.416660000013338
X: 0.9999986693954022, Y: -5.416660000013338
X: 0.9999986706830769, Y: -5.416660000013337
X: 0.9999986661156062, Y: -5.416660000013337

16 поколение, 5 лучших:
X: 0.9999986707271369, Y: -5.416660000013338
X: 0.9999986693950167, Y: -5.416660000013338
X: 0.99999866939526, Y: -5.416660000013338
X: 0.9999986693949083, Y: -5.416660000013338
X: 0.9999986693954022, Y: -5.416660000013338

Результат работы кнопки помощь

Помощь

Что такое генетический алгоритм?

Алгоритм представляет собой метод, отражающий естественную эволюцию методов решения проблем, и в первую очередь задач оптимизации. Генетические алгоритмы – это процедуры поиска, основанные на механизмах естественного отбора и наследования. В них используется эволюционный принцип выживания наиболее приспособленных особей. Они отличаются от традиционных методов оптимизации несколькими базовыми элементами. В частности, генетические алгоритмы:

1. Обрабатывают не значения параметров самой задачи, а их закодированную форму;
2. Осуществляют поиск решения исходя не из единственной точки, а из некоторой популяции;
3. Используют только целевую функцию, а не ее производные либо иную дополнительную информацию;
4. Применяют вероятностные, а не детерминированные правила выбора.

Генетические алгоритмы позволяют решать задачи прогнозирования, классификации, поиска оптимальных вариантов, и совершенно не зависимы в тех случаях, когда в обычных условиях решение задачи основано на интуиции, а не на строгом (в математическом смысле) ее описании.

Основные понятия генетических алгоритмов

Популяция – это конечное множество особей.

Особи, входящие в популяцию, в генетических алгоритмах представляются хромосомами с закодированным в них множествами параметров задачи, т.е. решений, которые иначе называются точками в пространстве поиска (search