Разработка обучающего модуля для работы с генетическими алгоритмами

Выполнила: ст.гр. ИТ-51
Пискулина М. С.
Дипломный руководитедь:
Катковская К. В.

Задачи программного модуля

1 Возможность реализации различных видов генетических алгоритмов;

2 Возможность выбора тематики различных классов задач;

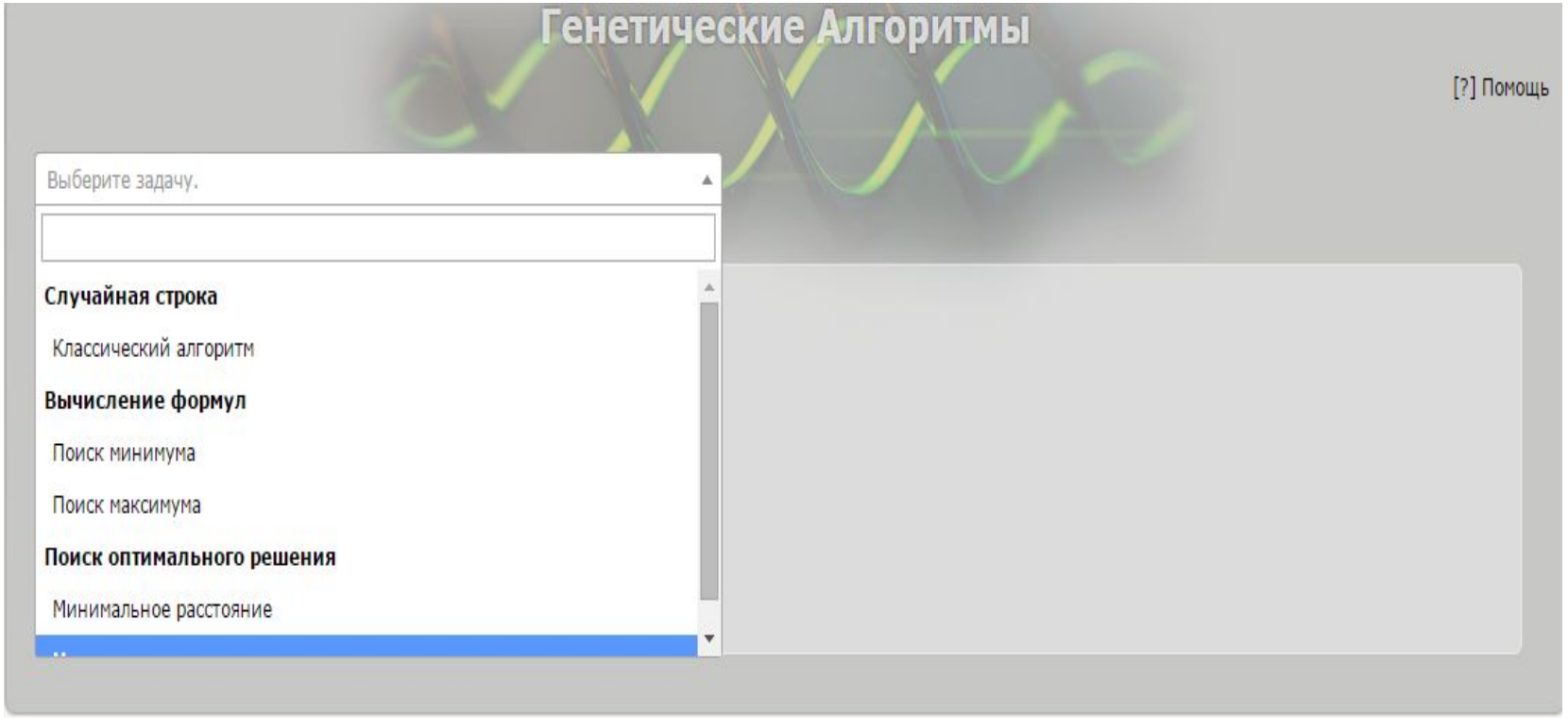
3 Наглядное интерактивное и динамическое представление этапов решения задач, в зависимости от выбранного алгоритма;

4 Наличие справки по использованию модуля.

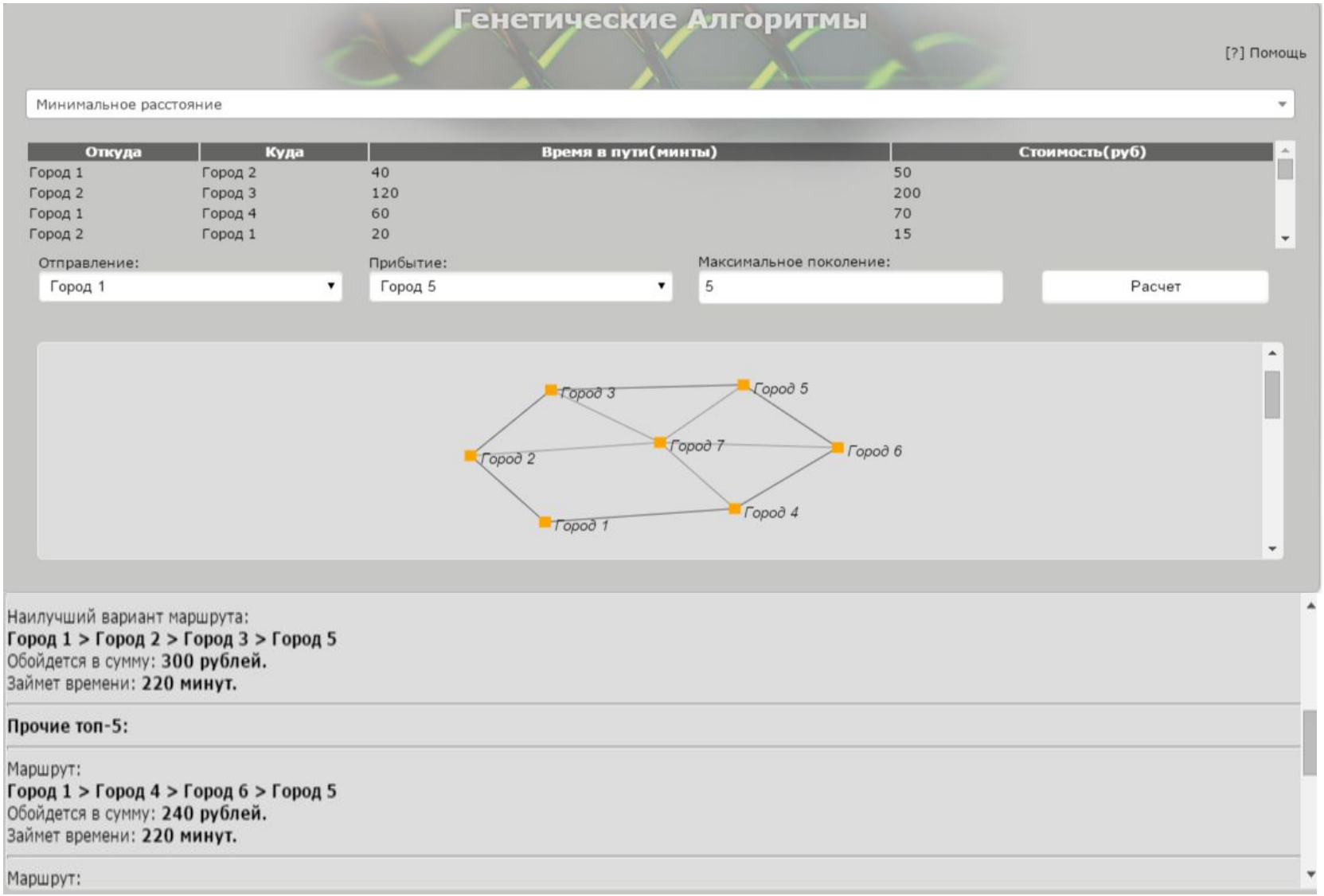
	ИС «GenSearch»	Flex Tool	3Genetic	ECL
Язык программирования	Delphi	Matlab	Java	Java
Доступ	Платная	Платная	Платная	Бесплатная
Интерфейс	Текстовый	Текстовый	Графический	Графический
Задачи	Осуществление генетического поиска, обеспечение настроек и управление всеми его элементами.	Выбор ГА, выбор метода 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

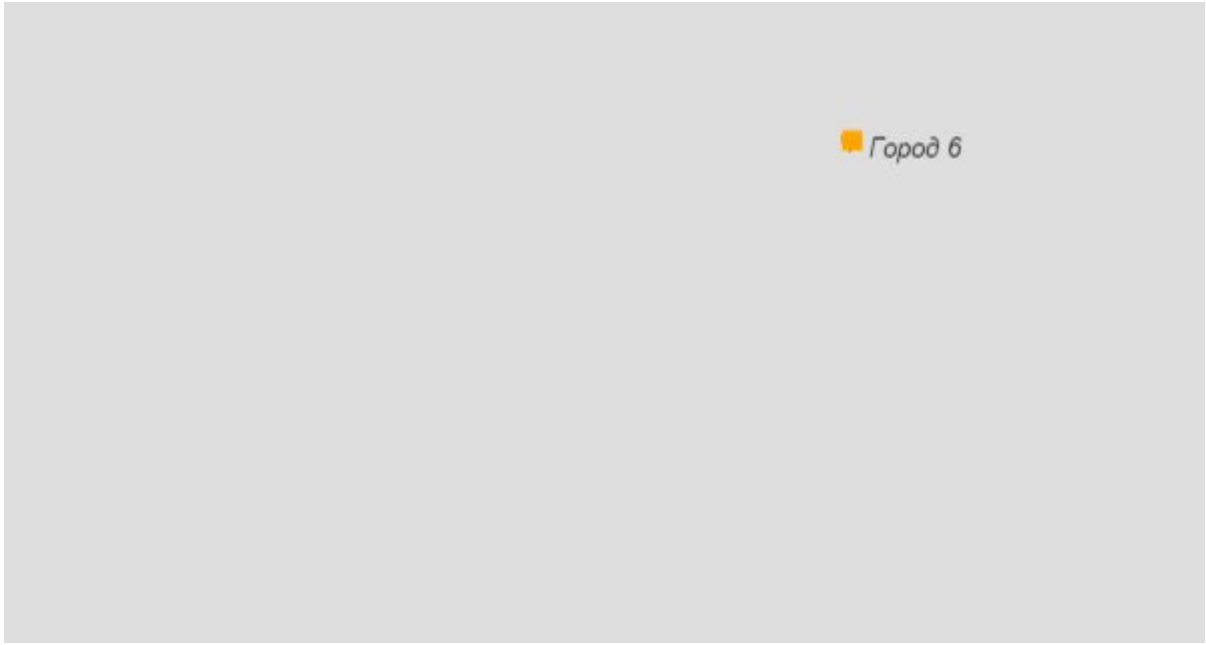
Внешний вид программного модуля для решения задач

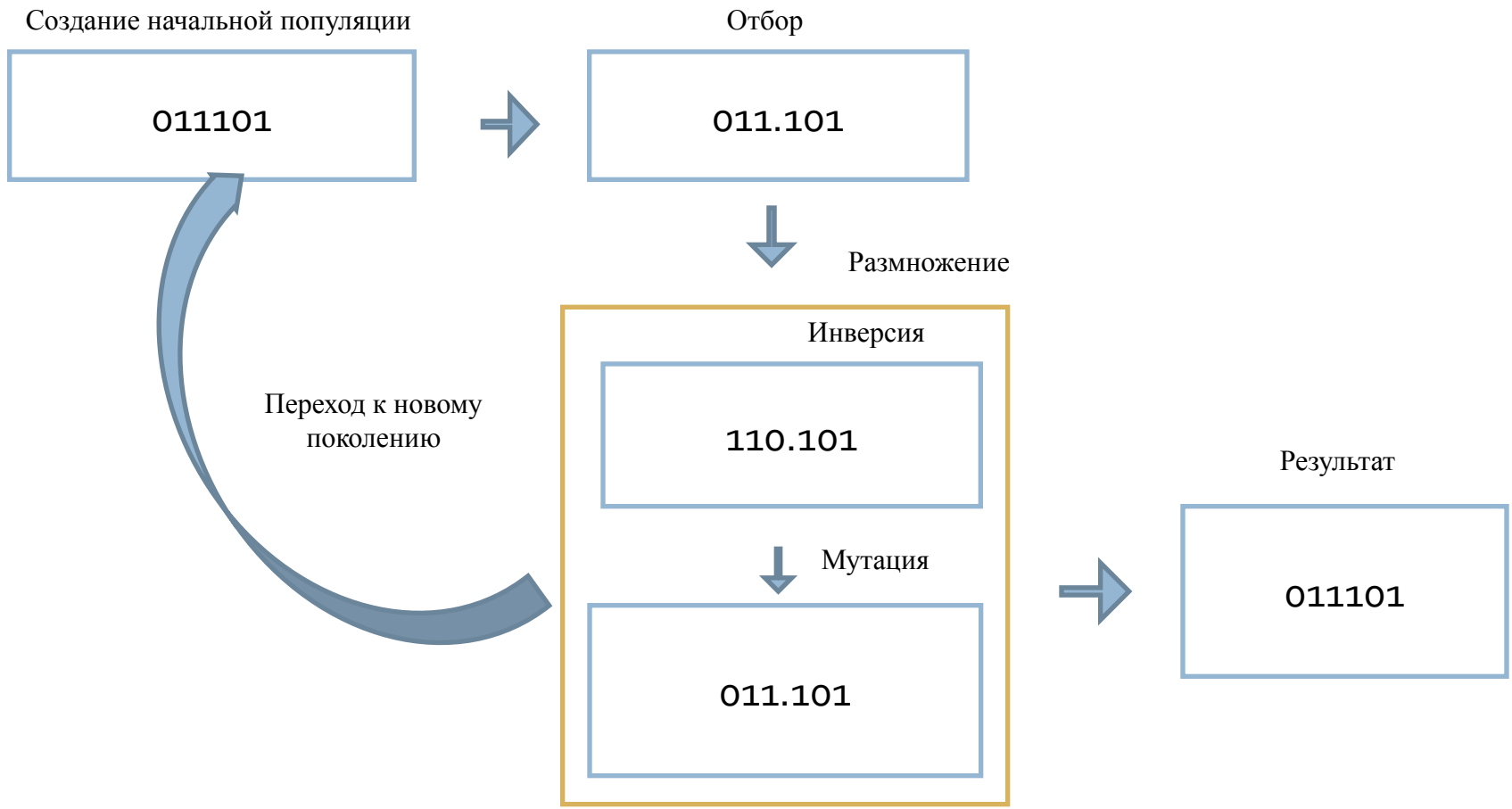


Нахождение минимального расстояния между городами



Пример динамичности программы




Создание начальной популяции
011101
Отбор
011.101
Размножение
Инверсия
110.101
Мутация
011.101
Переход к новому поколению
Результат
011101

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   $.grep(geoDATA, function(n){ return
n.from == fromCityName;
}).map(function(n){
9   //Запоминаем в wayIdx
10  wayIdx.push([$(geoDATA).index(n)]);
11  });
12  //Записываем это как новый геном
13  this.genome = wayIdx;
14  return true;
15  } else {
16  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 найденый маршрут
14  var asa = $(fitness).map(function
(i, elem){
15  //Соеденяем через запятую
16  return elem.join(',');
17  }).get();
18  var newFitness = [];
19  //Каждый найденый 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;}
```

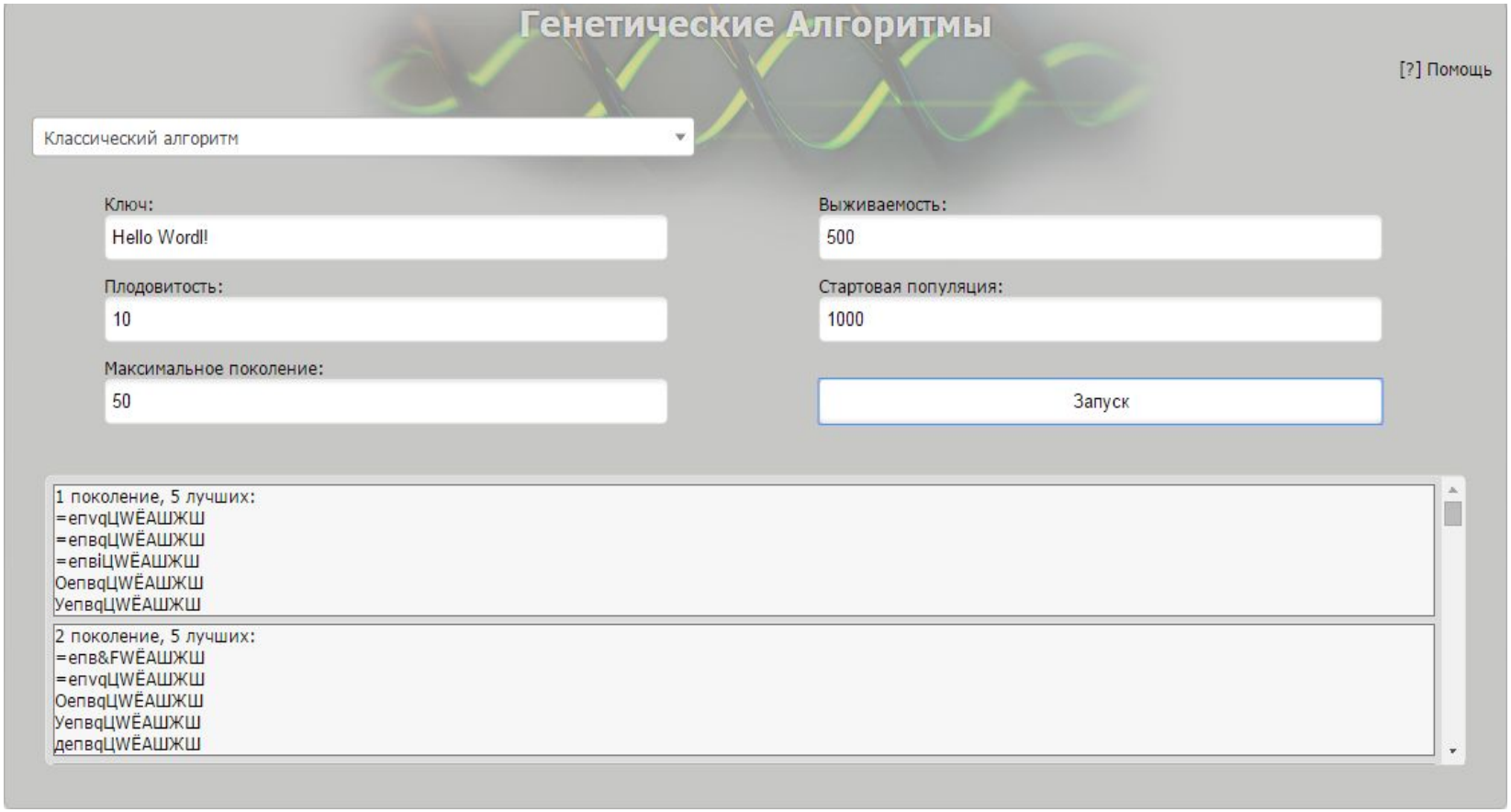
Размножение

```
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   //Ищим новые варианты путей
10  var newWays =
geneticTSP.getWaysByCheckpoint(geoDATA[lastElem]
.to);
11  //Все найденые 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  }
```

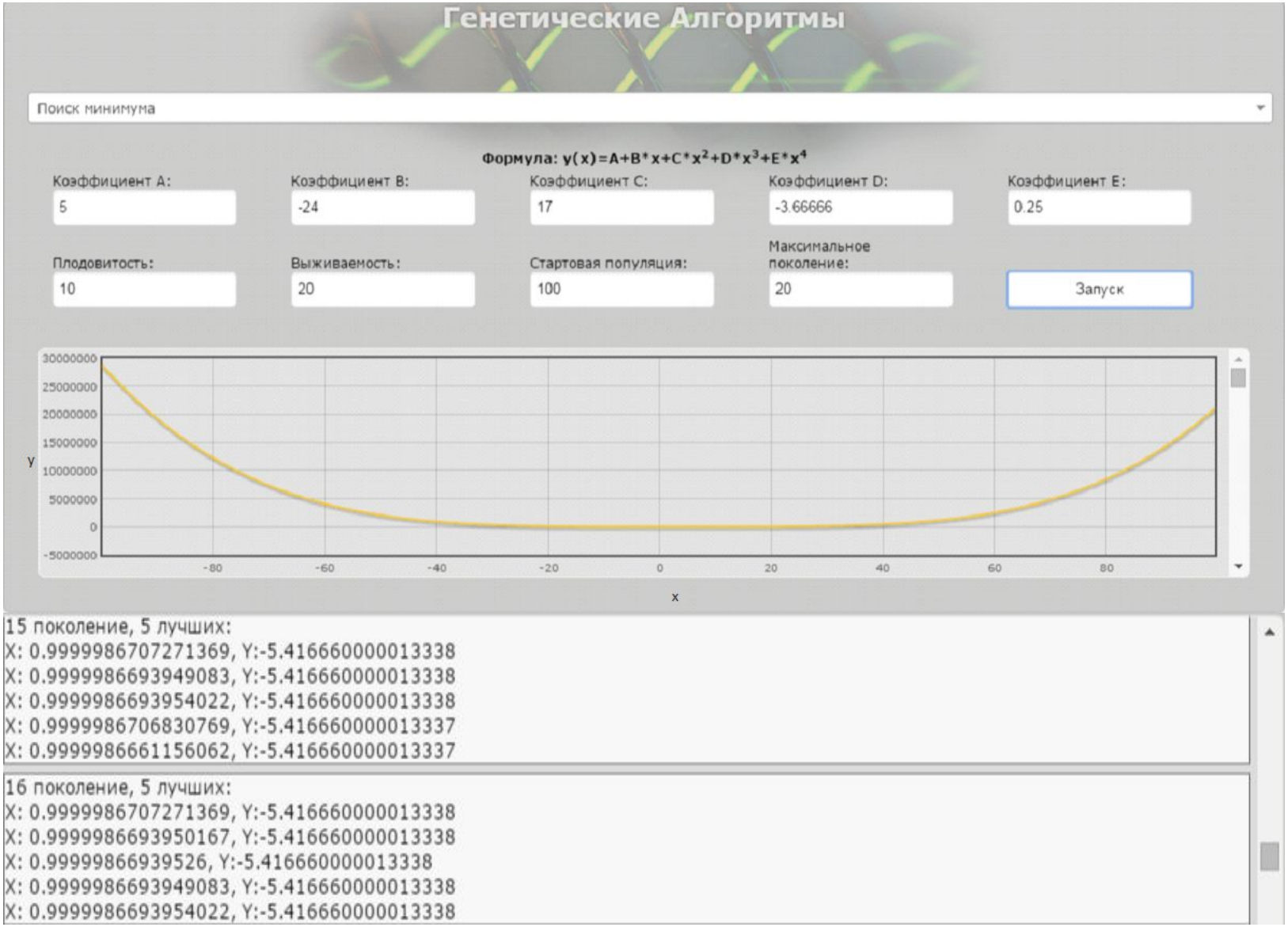
Результат

```
1   ...
2   $('.frame .output').html(html);
3   //записываем вывод результата в
окно с классом .output
4   sys.renderer =
Renderer("#"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  ...
```

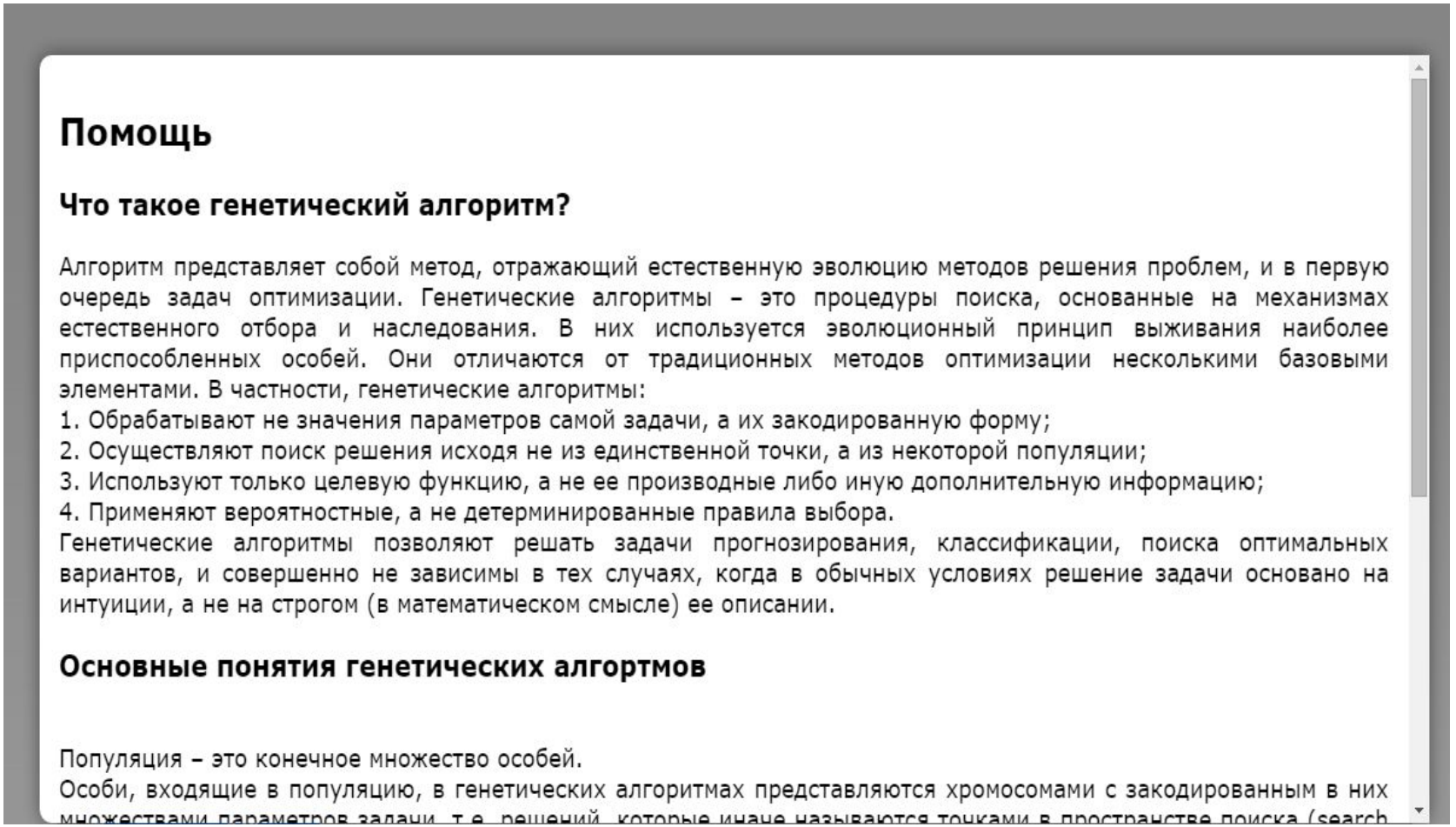
Реализация задачи «Случайная строка»



Нахождение «Минимума функции»



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 алгортмов

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