МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
(МИНОБРНАУКИ)

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»
(СПбГУ ИТМО)

УТВЕРЖДАЮ
Ректор СПбГУ ИТМО,
докт. техн. наук, профессор
В. Н. Васильев

____ _____ 2008 г.

ПРОГРАММНОЕ СРЕДСТВО 3GENETIC

ПРОГРАММНЫЙ МОДУЛЬ GA.SIMPLE
ТЕКСТ ПРОГРАММЫ

ЛИСТ УТВЕРЖДЕНИЯ

7.190.00001-01 12 05-ЛУ

Декан факультета «Информационные
технологии и программирование»
докт. техн. наук, профессор
_____ В. Г. Парфенов

Руководитель темы
заведующий кафедрой «Технологии программирования»,
докт. техн. наук, профессор
_____ А. А. Шалыто

2008

Инв.N подл.	Подп. и дата	Взам.инв.N	Инв.N дубл.	Подп. и дата

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
(МИНОБРНАУКИ)

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ
И ОПТИКИ»
(СПбГУ ИТМО)

УТВЕРЖДЕНО
7.190.00001-01 12 05-ЛУ

ПРОГРАММНОЕ СРЕДСТВО 3GENETIC

ПРОГРАММНЫЙ МОДУЛЬ GA.SIMPLE
ТЕКСТ ПРОГРАММЫ

7.190.00001-01 12 05

Листов 7

Инв. N подл.	Подп. и дата	Взам. инв. N	Инв. N дубл.	Подп. и дата

2008

АННОТАЦИЯ

В данном документе приводится текст модуля GA.simple программного средства 3GENETIC, реализующего классический вариант генетического алгоритма.

СОДЕРЖАНИЕ

1. МОДУЛЬ GA.SIMPLE

Модуль GA.simple является подключаемой частью программного средства 3GENETIC и представляет реализацию классического варианта генетического алгоритма. Исходный текст модуля хранится в 2-х файлах:

1. ga/simple/SimpleGA.java – реализация классического генетического алгоритма;
2. ga/simple/SimpleGALoader.java – загрузчик simpleGA.

1.1. Пакет ga.simple

1.1.1. ga/simple/SimpleGA.java

```
package ga.simple;

import java.util.*;

public class SimpleGA implements GA {

   private List<Individual> generation;

   private final int sizeElite;

   private final double probabilityMutation;

   private final IndividualFactory factory;

   private final Random r;

   @Override
   public void nextGeneration() {
      int size = generation.size();
      List<Individual> newGeneration = new ArrayList<Individual>(size);
      newGeneration.addAll(generation.subList(0, sizeElite));
      for (int i = 0; i < (size - sizeElite) / 2; i++) {
         Individual a1, a2;
         a1 = generation.get(r.nextInt(size));
         a2 = generation.get(r.nextInt(size));
         Individual p = (a1.compareTo(a2) < 0) ? a1 : a2;
         a1 = generation.get(r.nextInt(size));
         a2 = generation.get(r.nextInt(size));
         Individual[] s = p.crossover((a1.compareTo(a2) < 0) ? a1 : a2, r);
         newGeneration.add(s[0]);
         newGeneration.add(s[1]);
      }
      if (newGeneration.size() < size) {
         newGeneration.add(generation.get(r.nextInt(size)).mutate(r));
      }
      for (int i = 0; i < size; i++) {
         if (r.nextDouble() < probabilityMutation) {
            newGeneration.set(i, newGeneration.get(i).mutate(r));
         }
      }
```

```
        generation = newGeneration;
        Collections.sort(generation);
    }

    @Override
    public List<Individual> getGeneration() {
        return generation;
    }

    public SimpleGA(int sizeGeneration, double sizeElite, double probabilityMutation, IndividualFactory
            factory) {
        this.sizeElite = (int) (sizeElite * sizeGeneration);
        this.probabilityMutation = probabilityMutation;
        generation = new ArrayList<Individual>(sizeGeneration);
        for (int i = 0; i < sizeGeneration; i++) {
            generation.add(factory.randomIndividual());
        }
        /*for (Individual i : generation) {
            System.out.println(i.fitness());
        }*/
        Collections.sort(generation);
        this.factory = factory;
        r = new Random();
    }

    @Override
    public void bigMutation() {
        for (int i = 0; i < generation.size(); i++) {
            generation.set(i, factory.randomIndividual());
        }
        Collections.sort(generation);
    }

    @Override
    public Individual getBest() {
        return generation.get(0);
    }

    public void setGeneration(List<Individual> newGeneration) {
        generation.clear();
        generation.addAll(newGeneration);
    }
}
```

1.1.2. ga/simple/SimpleGALoader.java

```
package ga.simple;

import laboratory.common.Loader;
import laboratory.common.ga.GA;
import laboratory.common.ga.IndividualFactory;
import laboratory.util.Parser;
```

```
import java.io.IOException;
import java.util.Properties;
import java.util.jar.JarEntry;
import java.util.jar.JarFile;

public class SimpleGALoader implements Loader<GA> {

   private final Properties properties;

   public GA load(Object... args){
      return new SimpleGA(properties.getInt("sizeGeneration"),
            properties.getDouble("sizeElite"), properties.getDouble("probabilityMutation"),
            (IndividualFactory) args[0]);
   }

   public SimpleGALoader(JarFile file){
      Properties in = new Properties();
      try{
         JarEntry ent = new JarEntry("simpleGA.conf");
         in.load(file.getInputStream(ent));
      }catch (IOException e){
         e.printStackTrace();
      }
      properties = new Properties(in);
   }

   public Properties getProperties(){
      return properties;
   }

}
```

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Номера листов (страниц)					*Всего листов (страниц) в докум.*	*№ докумен-та*	*Входящий № сопрово-дительного докум. и дата*	*Подп.*	*Дата*
Изм.	*Измененных*	*Замененных*	*Новых*	*Аннулированных*					