Программное средство для изучения алгоритмов искусственного интеллекта «Виртуальная лаборатория 3Genetic»

Давыдов А. А., Соколов Д. С.
Научный руководитель: Царев Ф. Н.

8 октября 2010 г.

Содержание

1 Введение **3**

2 Общие правила **3**

3 Плагин, содержащий описание задачи **4**

4 Плагин, содержащий представление особи **4**

5 Плагин, содержащий схему генетического алгоритма **5**

6 Пример запуска виртуальной лаборатории **6**

1 Введение

Все упоминаемые ниже интерфейсы содержатся в файле common.jar, некоторые необязательные, но, возможно, полезные, при написании собственного модуля классы, содержатся в файле util.jar.

2 Общие правила

Создание плагина заключается в реализации соответствующего (*основного*) интерфейса и сборке jar-архива. Для подключения плагина манифест полученного архива должен содержать атрибут Main-Class, который в свою очередь должен ссылаться на класс реализующий один из интерфейсов, наследников PluginLoader.

```
public interface PluginLoader extends Nameable, Descriptable,
        Configurable {
    public Writer getConfigWriter(File file)
            throws FileNotFoundException;
}
public interface Nameable {
    public String getName();
}
public interface Configurable {
    public JDialog getConfigDialog(JFrame owner);
}
public interface Descriptable {
    public String getDescription();
}
```

- getConfigWriter — метод должен сохранять конфигурацию плагина в файл, возвращает открытый поток вывода в записываемый файл;
- getName — метод должен возвращать название плагина;
- getConfigDialog — метод должен возвращать диаловое окно с конфигурационными параметрами плагина.
- getDescription — метод должен возвращать текстовое описание плагина.

Класс, реализующий этот наследника данного интерфейс, должен иметь конструктор, от объекта класса (JarFile) (исключение плагин особи). То есть от того архива, где хранится сам плагин.

3 Плагин, содержащий описание задачи

Основной интерфейс плагина — TaskLoader.

```
public interface TaskLoader extends PluginLoader {
}
```

Данный интерфейс не содержит дополнительных методов.

4 Плагин, содержащий представление особи

Основной интерфейс плагина — IndividualLoader.

```
public interface IndividualLoader<I extends Individual>
        extends PluginLoader {
    public List<IndividualFactory<I>> loadFactories();
    public List<Crossover<I>> loadCrossovers();
    public List<Mutation<I>> loadMutations();
    public List<Fitness<I>> loadFunctions();
    public String getTaskName();

}
```

Данный интерфейс содержит следующие методы:

- loadFactories — метод должен возвращать *фабрику* особей, а именно объект класса, реализующего интерфейс IndividualFactory<I>.

  ```
  public interface IndividualFactory<I extends Individual> {
      public I getIndividual();
  }
  ```

  Единственный метод этого интерфейса должен возвращать случайно сгенерированную особь (интерфейс Individual).

  ```
  public interface Individual {
      public double standardFitness();
  }
  ```

  Метод standardFitness должен возвращать фитнесс-функцию для задачи, котую решает данная особь.

- loadCrossovers — метод должен возращать список доступных операторов скрещивания.

```
public interface Crossover<I extends Individual> {
    public List<I> apply(List<I> parents);
}
```

Метод apply по списку родительких особей должен возвражать список потомков, отдельно отметим, что особи потомков должны быть **новыми** объектами, это требуется для корректной работы алгоритма.

- loadMutations. loadFunctions — методы аналогичны методу loadCrossovers, должны возращать список доступных операторов мутации и подсчета фитнесс-функции.

- getTaskName — метод должен возращать название задачи, которую решает данная особь.

5 Плагин, содержащий схему генетического алгоритма

Основной интерфейс плагина — AlgorithmLoader.

```
public interface AlgorithmLoader<I extends Individual> extends PluginL
    public Algorithm<I> loadAlgorithm(
        List<IndividualFactory<I>> factories, List<Crossover<I>> crosso
        List<Mutation<I>> mutations, List<Selection<I>> sel, List<Fitn
    public String getMessage();
}
```

- loadFactories — метод должен возвращать *схему генетического алгоритма*, а именно объект класса, реализующего интерфейс Algorithm.

```
public interface Algorithm<I extends Individual> {
    public void nextGeneration();
    public List<I> getGeneration();
    public void stop();
}
```

Данный интерфейс содержит следующие методы:

  - nextGeneration — переход к следующему поколению особей.

  - getGeneration — метод должен возвращать текущее поколение особей.
  - stop — в текущей версии не используется.

- getTaskName — метод должен возвращать название задачи, которую решает данная особь.

- getMessage — метод должен возвращать сообщать сообщение о наличии и правильности загрузки опрераторов отбора (Selection).

```
public interface Selection<I extends Individual> {
    public List<FitIndividual<I>> apply(
        List<FitIndividual<I>> population, int n);
}
```

  Единственный метод данного интерфейса должен возвращать список особей, напрямую переходящих в следующее поколение.

6 Пример запуска виртуальной лаборатории

Опишем процесс запуска виртуальной лаборатории на примере решения задачи об "умном муравье"с помощью канонического генетического алгоритма. Чтобы осуществить запуск необходимо выполнить действия, описанные ниже.

1. Убедиться в наличие соответствующих плагинов:
   - tasks/ant.jar — плагин задачи об "умном муравье";
   - algorithms/simple.jar — плагин канонического генетического алгоритма;
   - individuals/mealy.jar — особь, представляющая собой конечный автомат Мили, для задачи об "Умном муравье";
   - functors/max.jar — плагин для отображения графика функции максимальной функции приспособленности от номера поколения;
   - functors/mean.jar — плагин для отображения графика функции максимальной функции приспособленности от номера поколения;
   - visualizators/sAnt.jar — визуализатор для особи mealy.
2. Запускаем виртуальную лабораторию при помощи:

- 3genetic.bat — для операционной системы Windows;
- 3genetic.sh — для Unix-like операционных систем.

3. В появившемся окне (рис. 1) нажимаем на кнопку Next.

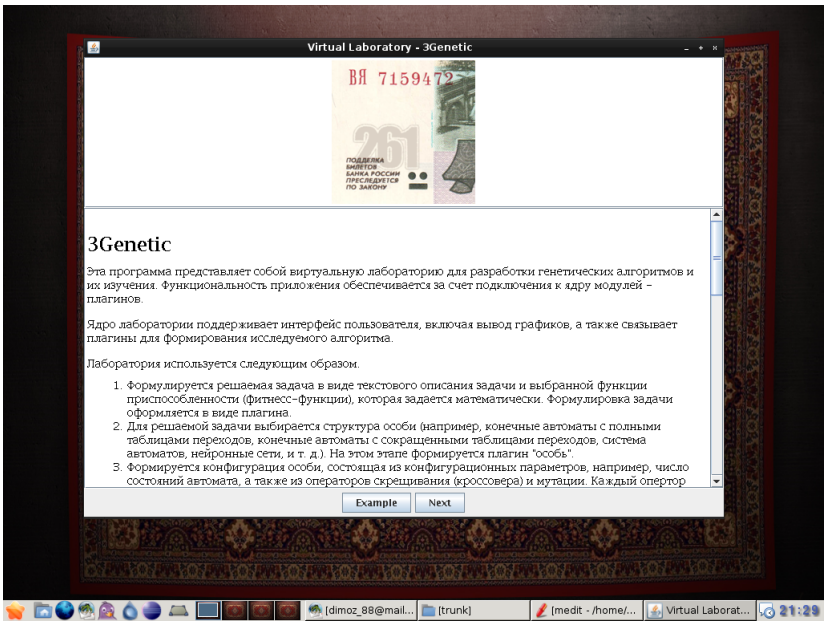

Рис. 1: Стартовое окно виртуальной лаборатории

4. Теперь на экране (рис. 2) можно видеть описание задач. В меню слева выбираем задачу Ant и нажимаем на кнопку Next.

5. В меню выбора особи (рис. 3) выбираем особь Mealy Automaton и нажимаем на кнопку Next.

6. В появившемся окне (рис. 4) находится меню выбора схемы генетического алгоритма, выбираем Canonical нажимаем на кнопку Load. Алгоритм готов к работе.

7. Для управления работой алгоритма используются соответствующие кнопки (рис. 5):

   - Start/Stop — запуск/остановка алгоритма;
   - Pause/Continue — постановка алгоритма на паузу/продолжение работы алгоритма после постановки на паузу.

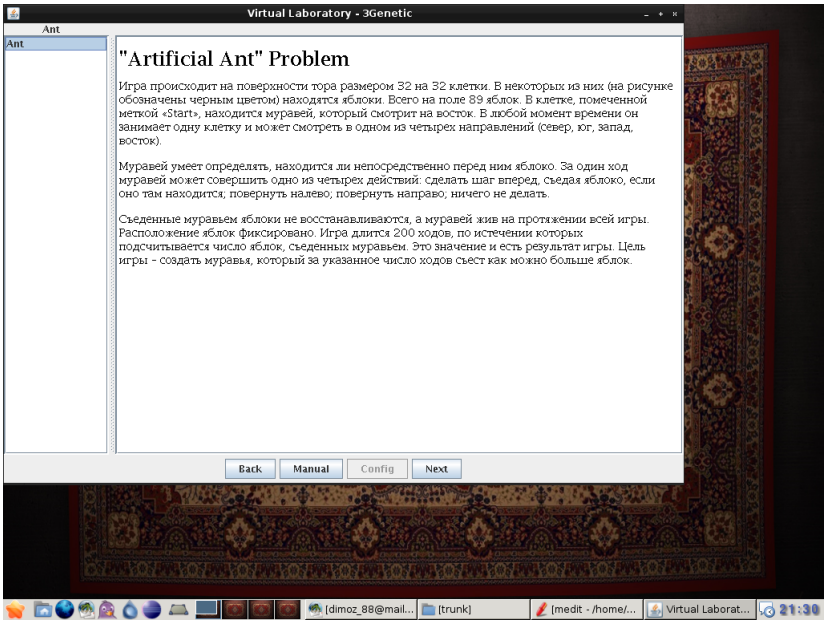

Рис. 2: Окно выбора задачи

8. Нажимаем кнопку Start, на экране появляются графики максимальной и средней фитнесс-функции от номера поколения (рис. 6). После получения желаемых значений нажимаем кнопку Pause.

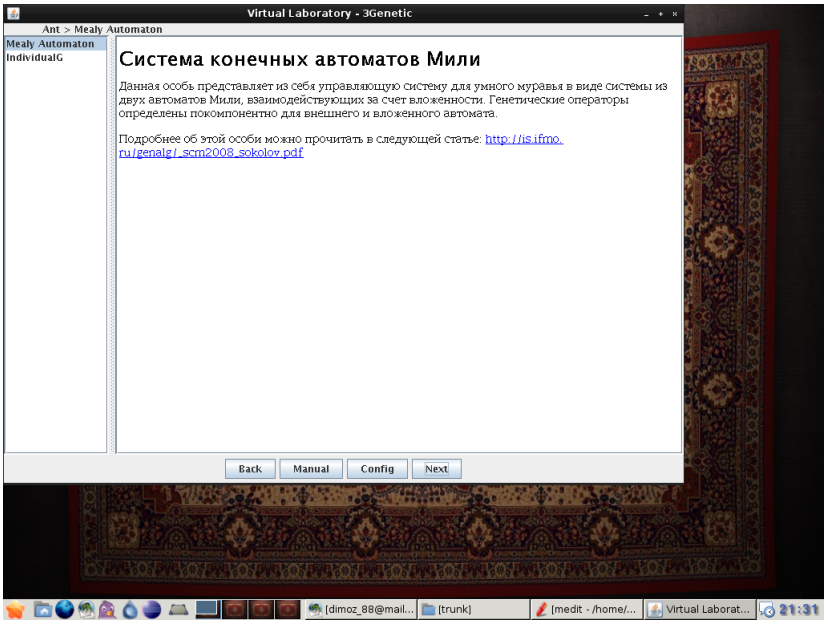

Рис. 3: Окно выбора задачи

9. При нажатии в меню на кнопку Show, появляется возможность просмотреть лучших особей каждого поколения и любую особь из текущего поколения. Нажимаем на кнопку Best individuals.

10. В появившемся окне выбираем особь, которую хотим посмотреть (рис. 7). Нажимаем на кнопку OK.

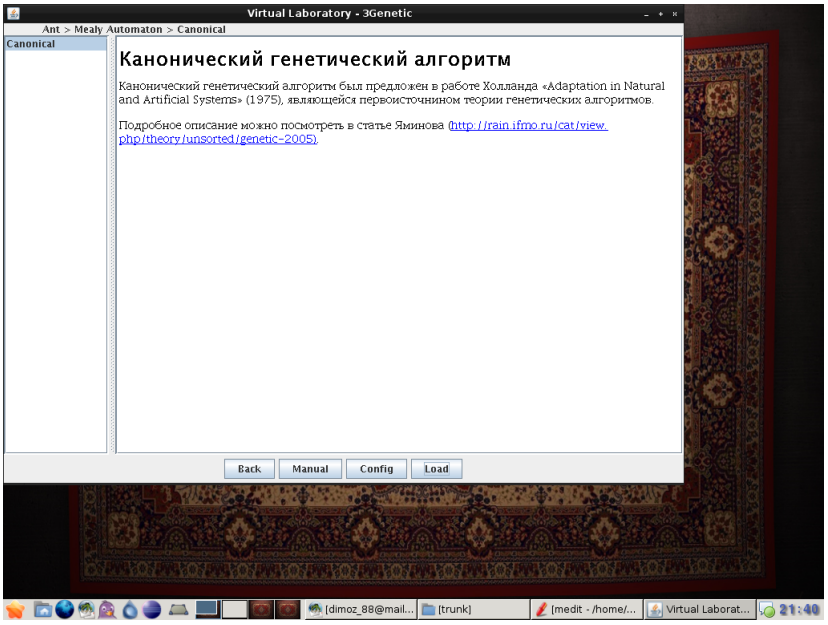

Рис. 4: Окно выбора схемы генетического алгоритма

11. Выбираем визуализатор Chebotareva's visualizator. Нажимаем на кнопку OK. Открывается окно в выбранной особью (рис. 8).

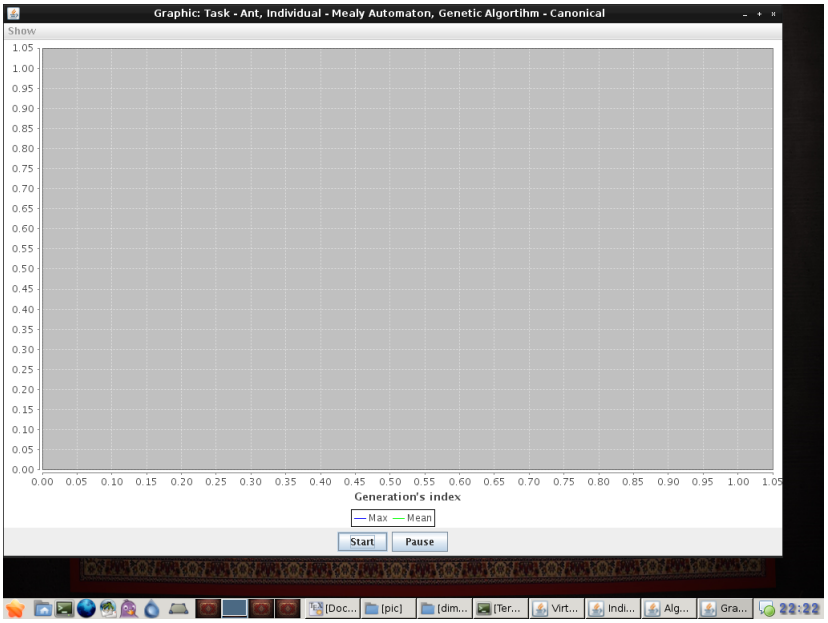

Рис. 5: Окно управления работой генетического алгоритма

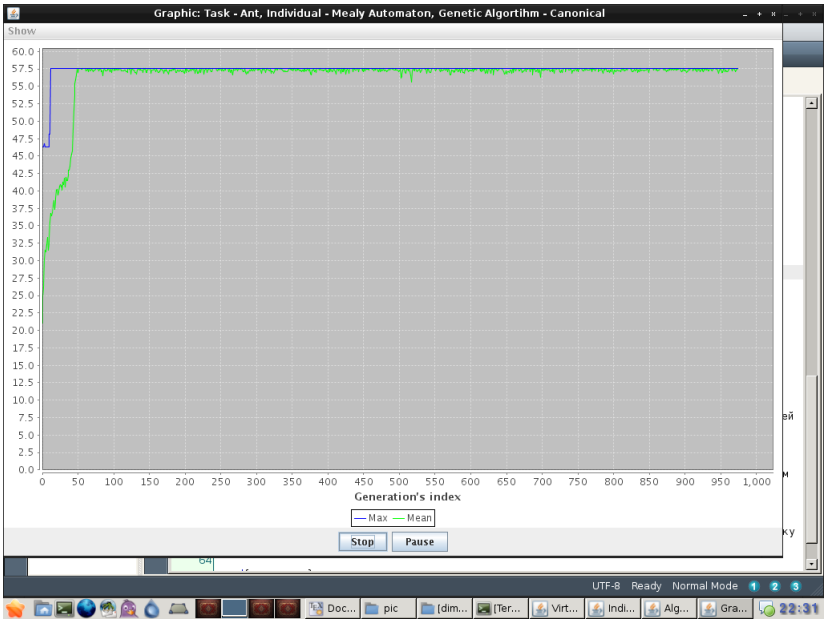

Рис. 6: Статистика работы генетического алгоритма

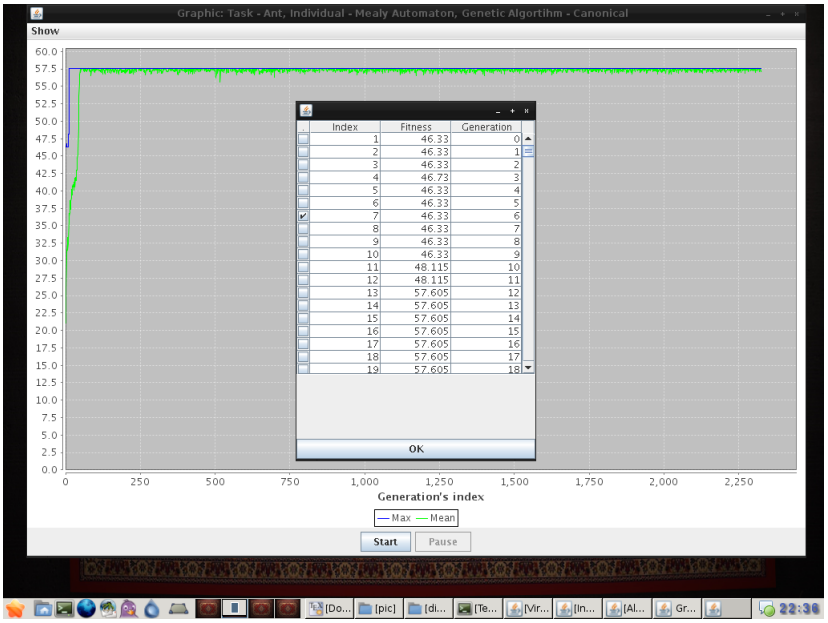

Рис. 7: Окно выбора особи для просмотра

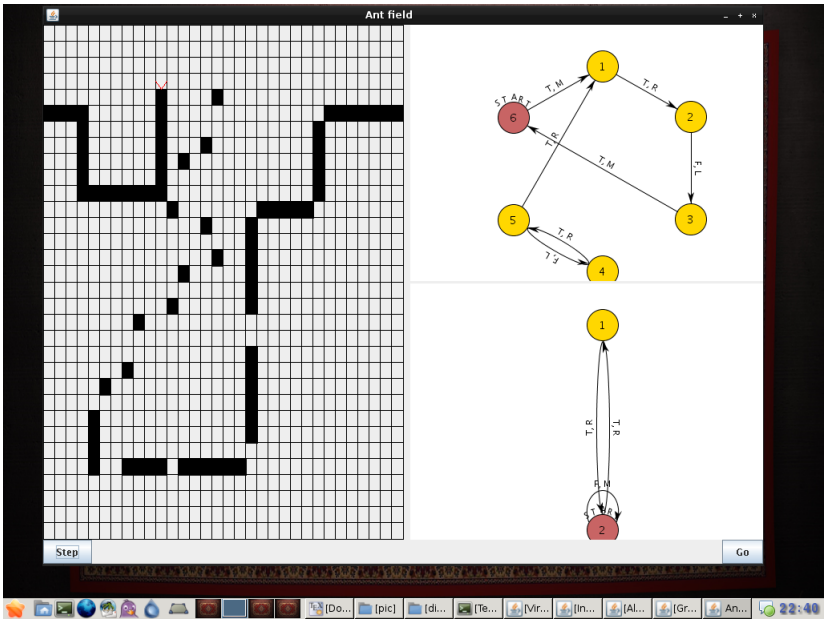

Рис. 8: Окно выбора особи для просмотра