Материал опубликован в сборнике докладов IV Международной научно-практической конференции «Современные информационные технологии и ИТ-образовние». М.: ИНТУИТ. РУ, МГУ. 2009, с. 222–227.

Виртуальная лаборатория обучения методам искусственного интеллекта для генерации управляющих конечных автоматов

А. С. Тяхти, А. А. Чебатуркин, Ф. Н. Царев, А. А. Шалыто,
Санкт-Петербургский государственный университет информационных технологий, механики и оптики, tyahti@rain.ifmo.ru

1. Введение

Парадигма автоматного программирования [1,2] была предложена в 1991 году в России. В рамках данной концепции программа рассматривается как набор конечных автоматов, объектов управления и поставщиков событий.

Зачастую построение конечных автоматов эвристическими методами является затруднительным. Поэтому для их генерации можно использовать автоматизированные методы, в том числе генетические алгоритмы и генетическое программирование [3].

Для обучения генетическому программированию ранее была создана виртуальная лаборатория для языка программирования Java [4]. Однако в указанной виртуальной лаборатории отсутствовала возможность применения других методов искусственного интеллекта таких, как, например, метод имитации отжига [5–9]. Эта техника оптимизации использует случайный поиск на основе аналогии с процессом образования в веществе кристаллической структуры с минимальной энергией, происходящем при охлаждении этого вещества.

На основании изложенного можно сделать вывод о том, что актуальной является разработка виртуальной лаборатории для обучения методам искусственного интеллекта. При этом важным требованием для такой виртуальной лаборатории является возможность не только самостоятельно разрабатывать методы и создавать новые задачи для их тестирования, но и сравнивать различные алгоритмы между собой, применительно к конкретным задачам, а также наглядно визуализировать полученные решения.

2. Основные положения

Виртуальная лаборатория GlOpt реализована на языке программирования C# на платформе *Microsoft.NET*. Она состоит из ядра и подключаемых модулей (плагинов), которые реализуют конкретные задачи и методы их решения.

Ядро представлено классом `Brain`, который ответственен за загрузку основных сущностей программного комплекса: задач, методов искусственного интеллекта и визуализаторов, а также выполняет функции распределения ресурсов между работающими алгоритмами.

Для описания задач используются абстрактные классы `Problem` и `Individual`, от которых наследуются классы, описывающие конкретные задачи, например задачу об «Умном муравье» [10]. Алгоритмы решения описываются классами `Algorithm` и `SearchOperator`, от которых в свою очередь наследуются классы, реализующие конкретные методы искусственного интеллекта, например, метод имитации отжига. Общая структура виртуальной лаборатории приведена на рис. 1.

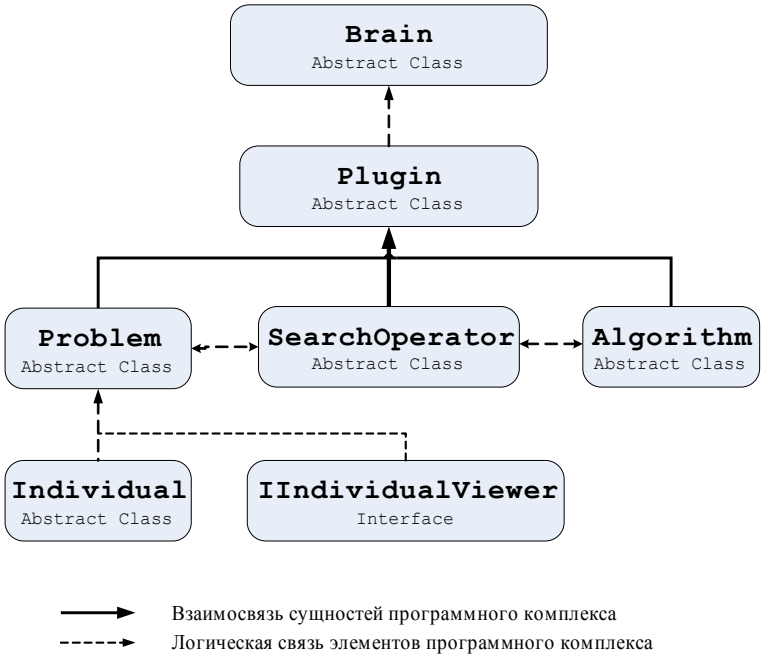

Рис. 1. Структура виртуальной лаборатории

Таким образом, в виртуальной лаборатории существует пять типов плагинов:

- рассматриваемые задачи (наследуется от абстрактного класса Problem);
- алгоритмы и методы искусственного интеллекта (наследуется от абстрактного класса `Algorithm`);
- методы, адаптирующие алгоритмы к конкретным задачам (наследуется от абстрактного класса `SearchOperator`);
- визуализаторы для задач (рис. 2);
- текстовые описания задач и алгоритмов.

Каждый плагин представляет собой dll-библиотеку. Особо отметим, что виртуальная лаборатория спроектирована таким образом, что любую задачу можно решать, используя каждый из реализованных методов искусственного интеллекта.

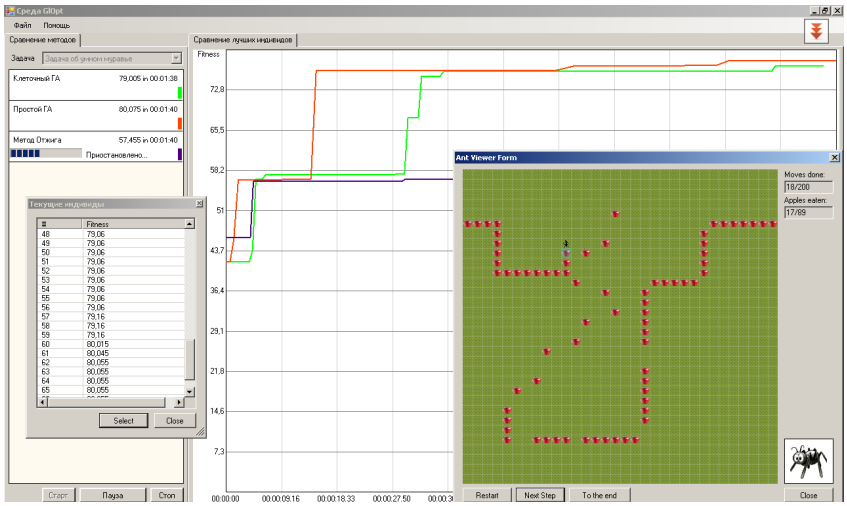

Рис. 2. Визуализатор для задачи об «Умном муравье»

В настоящее время программный комплекс виртуальной лаборатории включает в себя следующие плагины:

- задачи:
  - об «Умном муравье» (рис. 3);
  - о расстановке N ферзей на шахматной доске[5];

- методы искусственного интеллекта:
  - простой генетический алгоритм;
  - клеточный генетический алгоритм [3];
  - метод имитации отжига.
- визуализаторы для указанных задач;
- графики, позволяющие оценивать эффективность методов применительно к конкретным задачам;
- документация, в том числе html-описания решаемых задач и используемых для их решения методов.

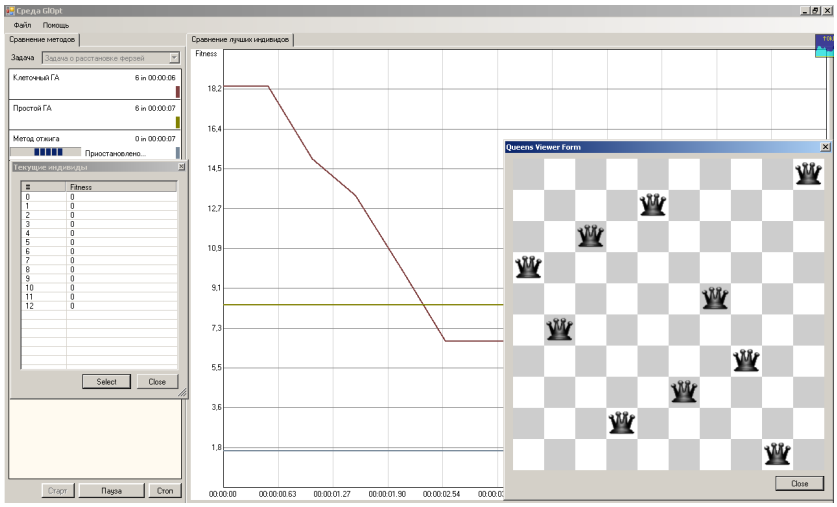

Рис. 2. Задача о расстановке N ферзей на шахматной доске

3. Создание плагинов

Опишем процесс создания плагинов на примере плагина задачи и алгоритма решения.

3.1. Создание плагина-задачи

Основные этапы при реализации новой задачи в виртуальной лаборатории GLOpt.

1. Реализация класса-наследника от абстрактного класса `Individual`, определяющего объект к которому в дальнейшем применяется алгоритм оптимизации. Так, например, в задаче о расстановке ферзей таким объектом

выступает шахматная доска с расставленными на ней ферзями.

2. Реализация класса-наследника от абстрактного класса `Problem`. В данном классе должен быть определен метод `OptimizationDirection`. Также необходимо определить метод `EvaluateIndividual`, возвращающий значение типа `double` – значение целевой функции для данного индивида.
3. Для реализации компоненты визуализации текущего решения требуется определить класс `Viewer`, наследуемый от класса `IndividualViewer`, и, в частности, определить метод `ViewIndividual`, вызывающий графическую форму, либо представляющий данные о решении в любом другом удобном для пользователя виде.
4. Библиотека (dll-файл) откомпилированного модуля должна находиться в папке plugins лаборатории.

3.2. **Создание плагина-алгоритма**

Приведем основные этапы разработки собственного подключаемого модуля, определяющего алгоритм.

1. Реализация алгоритма поиска оптимального решения – класса, наследованного от абстрактного класса `Algorithm`. Данный класс должен определять следующие методы.
    - `OptimizationDirection` – возвращает одну из двух именованных констант: `OptimizationDirection.Minimize` или `OptimizationDirection.Maximize`.
    - `Initialize` – описывает начальное состояние в алгоритме поиска решения.
    - `NextIteration` – описывает действия, происходящие на очередной итерации алгоритма.
    - В переменной `BestIndividual` типа `Individual` должна содержаться актуальная информация об объекте типа `Individual` с лучшей на данной итерации алгоритма целевой функцией.

2. Реализация интерфейса `SearchOperator`, наследованного от интерфейса `ICreateOperator`. В данном интерфейсе должен быть определен метод `Create`, возвращающий объект типа `Individual` (абстрактный класс, для каждой задачи используется своя реализация). Также в `SearchOperator` реализуются все необходимые методы для работы с объектами `Individual` – например, операции мутации и скрещивания.
3. Библиотеки *.dll откомпилированного модуля должны находится в папке plugins лаборатории.

4. Сравнение виртуальных лабораторий

В таблице приведены основные сравнительные характеристики настоящей виртуальной лаборатории с ранее созданной лабораторией на языке Java [4].

Таблица. Сравнительные характеристики лабораторий

Критерий сравнения	**Лаборатория GlOpt**	**Лаборатория на Java**
Язык программирования	C#	Java
Поддержка методов искусственного интеллекта	Генетическое программирование, метод имитации отжига Поддержка добавления новых методов в качестве плагинов.	Только генетическое программирование.
Возможность одновременного запуска алгоритмов	Поддерживается. Результат работы отображается в виде сводного графика.	Не поддерживается.
Встроенная документация	В формате HTML.	В формате HTML.

5. Заключение

Важной особенностью виртуальной лаборатории GlOpt является возможность применения реализуемых алгоритмов и методов для любой из рассматриваемых задач. Наглядное сравнение результатов работы алгоритмов, удобство анализа их эффективности при влиянии тех или иных факторов настройки обеспечивается наличием сводных

графиков работы методов, средств визуализации, доступа к базовой информации о задачах и алгоритмах непосредственно из самой виртуальной лаборатории.

Гибкая система плагинов, позволяющая реализовывать весь комплекс необходимой функциональности – от новых задач до визуализаторов к ним, призвана максимально упростить реализацию новых модулей, что облегчает понимание основ методов искусственного интеллекта, и в частности методов имитации отжига, генетических алгоритмов.

В будущем планируется продолжение работы по созданию максимально простой и понятной системы реализации дополнительных модулей. Также внимание будет сосредоточено на совершенствовании средств документирования задач и методов их решения.

Литература

1. Шалыто А.А. SWITCH технология. Алгоритмизация и программирование задач логического управления. СПб. Наука, 1998. http://is.ifmo.ru/books/switch/1
2. Поликарпова Н. И., Шалыто А. А. Автоматное программирование. СПб. Питер, 2009.
3. Koza J. Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, 1992.
4. Соколов Д.О., Давыдов А.А., Царев Ф.Н., Шалыто А.А. Виртуальная лаборатория обучения генетическому программированию для генерации управляющих конечных автоматов / Сборник трудов третьей Международной научно-практической конференции «Современные информационные технологи и ИТ-образование». М.: МАКС Пресс, 2008, с. 179 – 183. http://is.ifmo.ru/works/_2_93_davidov_sokolov.pdf
5. Ingber A.L. Simulating Annealing: Practice versus theory. Mathl. Comput. Modelling, 1993.
6. Ёлкин Д. И. Тяхти А. С. Метод отжига - СПбГУ ИТМО, 2008, http://rain.ifmo.ru/cat/view.php/theory/unsorted/ai-annealing-2008/article.pdf
7. Джонс М. Т. Программирование искусственного интеллекта в приложениях – М.: ДМК-Пресс, 2004
8. Лопатин А. Методы отжига. Электронный конспект – Крысталь Б. www.cs-seminar.spb.ru/reports/52.pdf
9. Орлянская И.В Современные подходы к построению методов глобальной оптимизации. http://zhurnal.ape.relarn.ru/articles/2002/189.pdf
10. Бедный Ю.Д., Шалыто А.А. Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей». http://is.ifmo.ru/works/_ant.pdf