УДК 004.4’242

ВИРТУАЛЬНАЯ ЛАБОРАТОРИЯ ОБУЧЕНИЯ МЕТОДАМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ ГЕНЕРАЦИИ УПРАВЛЯЮЩИХ КОНЕЧНЫХ АВТОМАТОВ

А.С. Тяхти

Описывается структура и возможности виртуальной лаборатории для обучения генетическому и автоматному программированию, реализованной на языке C#. Описываются основные этапы создания собственных подключаемых модулей лаборатории.

Ключевые слова: автоматное программирование, виртуальная лаборатория.

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. В связи с этим 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.

Основные положения

Виртуальная лаборатория GlOpt (сокращение от Global Optimization) 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 отвечает 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].

Класс Problem содержит метод OptimizationDirection, возвращающий информацию о направлениях оптимизации (min, max) и метод EvaluateIndividual, служащий для вычисления функции приспособленности у конкретной особи. Наследники этого класса должны реализовывать эти методы, а также методы, обеспечивающие доступ к базовой информации о задаче – названии, ее кратком описании. Класс Individual реализует представление индивидуальной для каждой задачи оценочной характеристики Fitness, описывает метод сравнения индивидов между собой.

Алгоритмы решения описываются классами Algorithm и SearchOperator, от которых, в свою очередь, наследуются классы, реализующие конкретные методы искусственного интеллекта, например, метод имитации отжига. Класс SearchOperator необходим для организации требуемых в задаче операций над объектами класса Individual. Он обеспечивает универсальность в применении методов решения к различным задачам. Так, например, для применения уже реализованного алгоритма к задаче об «Умном муравье» требуется определить методы мутации и скрещивания индивидов, а для метода имитации отжига – изменение уже найденного решения в соответствии с выбранным вероятностным распределением.

Общая структура виртуальной лаборатории приведена на рис. 1.

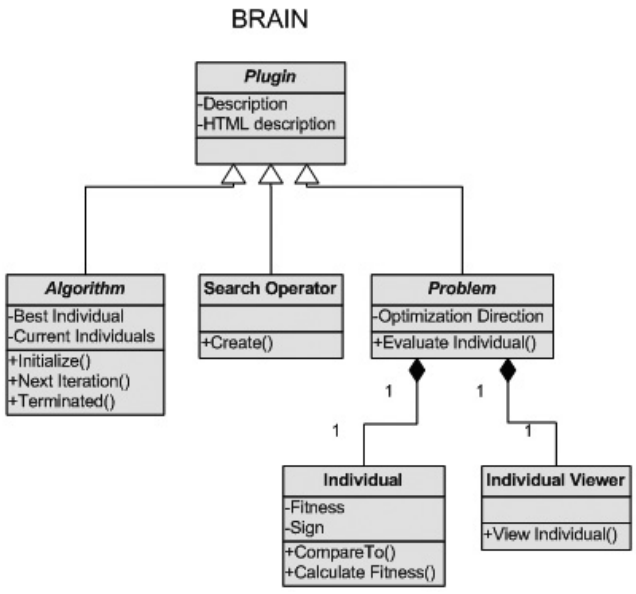

Рис. 1. Структура виртуальной лаборатории

Таким образом, в виртуальной лаборатории существует пять типов плагинов:

1. рассматриваемые задачи (наследуется от абстрактного класса Problem);
2. алгоритмы и методы искусственного интеллекта (наследуется от абстрактного класса Algorithm);
3. методы, адаптирующие алгоритмы к конкретным задачам (наследуется от абстрактного класса SearchOperator);
4. визуализаторы для задач (пример внешнего вида визуализатора приведен на рис. 2);
5. 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.

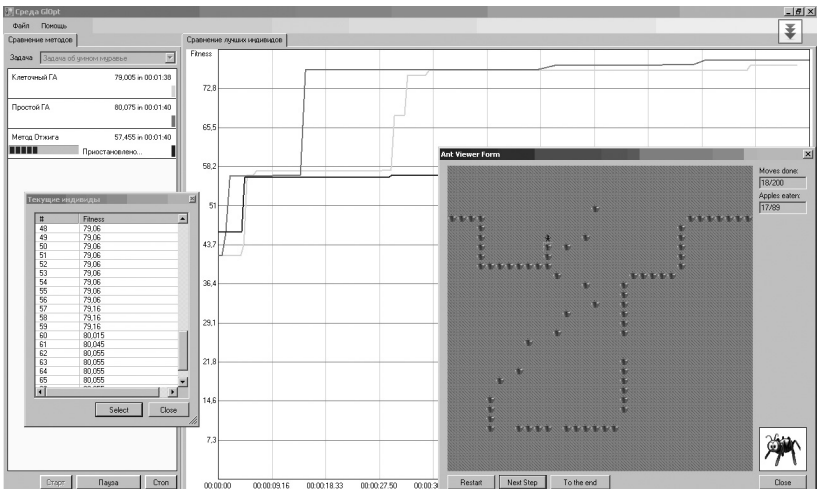

Рис. 2. Визуализатор для задачи об «Умном муравье»

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

- задачи:
  - об «Умном муравье»;
  - об «Умном муравье-3»;
  - о роботе, огибающем препятствия;
  - о расстановке *N* ферзей на шахматной доске (рис. 3);
- методы искусственного интеллекта:
  - генетические алгоритмы – классический, клеточный, островной;
  - методы имитации отжига – больцмановский, Коши, тушения;
  - эволюционные стратегии;
  - метод оптимизации роем частиц;
  - пчелиный алгоритм;
- 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.

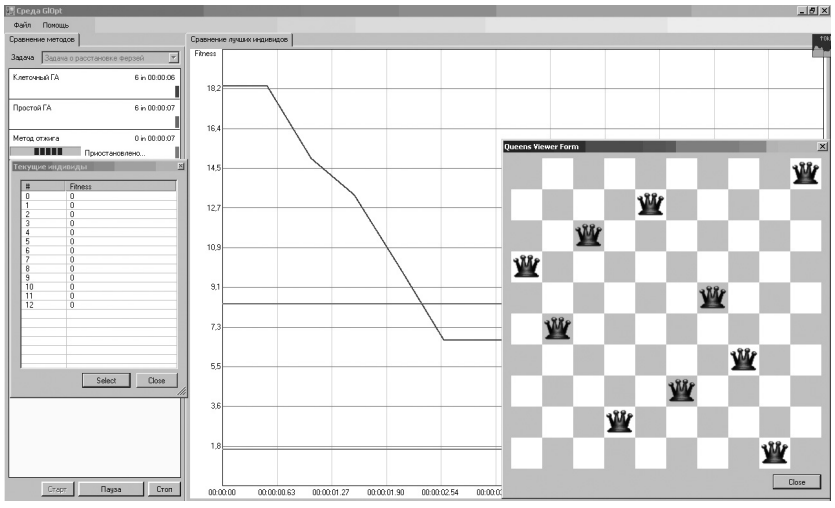

Рис. 3. Задача о расстановке *N* ферзей на шахматной доске

Создание плагинов

Проектирование модулей предполагает изучение структуры уже реализованных методов оптимизации и построение на их основе собственных. Ниже приведены базовые рекомендации, которым необходимо следовать при разработке собственных модулей, определяющих алгоритм решения. Процесс создания плагинов описывается на примере создания плагина-задачи и алгоритма решения.

Создание плагина-задачи. Приведем основные этапы при реализации новой задачи в виртуальной лаборатории GlOpt.

- 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.
- 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.
- 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.
- Библиотека (dll-файл) откомпилированного модуля должна находиться в папке plugins лаборатории.

Создание плагина-алгоритма. Приведем основные этапы разработки собственного подключаемого модуля, определяющего алгоритм.

- 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.
- 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.
- Библиотеки *.dll откомпилированного модуля должны находиться в папке plugins лаборатории.

Сравнение виртуальных лабораторий

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

Критерий сравнения	Лаборатория GlOpt	Лаборатория на Java
Язык программирования	C#	Java
Поддержка задач	Задачи об «Умном муравье», о расстановке ферзей Поддержка добавления новых задач в качестве плагинов	Только задача об «Умном муравье»
Возможность одновременного запуска алгоритмов	Поддерживается Результат работы отображается в виде сводного графика	Не поддерживается
Встроенная документация	В формате html	Отсутствует

Таблица. Сравнительные характеристики лабораторий

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.

Параллельно с выполнением настоящей работы А. Цветковым [11] велась работа по созданию виртуальной web-лаборатории, с помощью которой стало возможным проводить исследование реализованных методов и задач на удаленном сервере, вне зависимости от наличия на конкретной локальной машине того или иного программного обеспечения.

Исследование выполнено по Федеральной целевой программе «Научные и научно-педагогические кадры инновационной России на 2009–2013 годы» в рамках государственного контракта П2236 от 11 ноября 2009 года.

Литература

1. Шалыто А.А. Switch-технология. Алгоритмизация и программирование задач логического управления. – СПб: Наука, 1998. – 628 с.
2. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб: Питер, 2009 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/books/_book.pdf, своб.
3. Koza J.R. Genetic programming: On the Programming of Computers by Means of Natural Selection. – Cambridge: 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 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/works/_2_93_davidov_sokolov.pdf, своб.
5. Ingber A.L. Simulating Annealing: Practice versus theory. – Mathl. Comput. Modelling, 1993.
6. Ёлкин Д.И. Тяхти А.С. Метод отжига. – СПб: СПбГУ ИТМО, 2008 [Электронный ресурс]. – Режим доступа: http://rain.ifmo.ru/cat/view.php/theory/unsorted/ai-annealing-2008/article.pdf, своб.
7. Джонс М.Т. Программирование искусственного интеллекта в приложениях – М.: ДМК-Пресс, 2004.
8. Лопатин А. Методы отжига. Электронный конспект – Крысталь Б. [Электронный ресурс]. – Режим доступа: www.cs-seminar.spb.ru/reports/52.pdf, своб.
9. Орлянская И.В. Современные подходы к построению методов глобальной оптимизации [Электронный ресурс]. – Режим доступа: http://zhurnal.ape.relarn.ru/articles/2002/189.pdf, своб.
10. Бедный Ю.Д., Шалыто А.А. Применение генетических алгоритмов для построения автоматов в задаче «Умный муравей». – СПбГУ ИТМО, 2007 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/works/ant, свободный. Яз. рус. (дата обращения 09.02.2011).
11. Цветков А.А. Сравнение поведенческих и эволюционных алгоритмов построения управляющих конечных автоматов. Бакалаврская работа. – СПб: СПбГУ ИТМО, 2010.

Тяхти Александр Сергеевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, студент, tyahti@gmail.com