Материал опубликован в сборнике докладов XI Международной конференции по мягким вычислениям и измерениям (SCM'2008). СПб.: СПбГЭТУ. 2008, с. 261-265.

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ ДЛЯ СОЗДАНИЯ СИСТЕМЫ УПРАВЛЕНИЯ ТАНКОМ В ИГРЕ *ROBOCODE*

Ю. Д. Бедный, А. А. Шалыто

Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Abstract. In this paper, we propose a simple while efficient approach for evolving tanks automatically for the *Robocode* game by means of genetic algorithms. Then the suggested approach is generalized to a more powerful automata-based method. Using these techniques we were able to evolve tanks that beat some competitive (yet not the best) hand-coded adversaries.

Введение

В работе предлагается метод автоматизации построения системы управления танком для популярной компьютерной игры *Robocode* [1]. Этот метод позволяет на основе использования генетических алгоритмов [2, 3] строить достаточно простые системы управления танком.

Танки, создаваемые предлагаемым методом, должны побеждать в индивидуальном сражении каждого **заранее выбранного соперника** из тестового набора, который включает в себя танки, поставляемые вместе с игрой, а также танк `newCynic.Cynical` [4]. Для каждого танка из тестового набора с помощью предлагаемого метода генерируется танк, который его побеждает. Предлагаемый метод позволяет генерировать танки, системы управления которых, построены как с применением автоматного подхода [5], так и без его использования.

Обзор

Известно большое число танков для рассматриваемой игры [6]. Большинство из них создано вручную. Существуют также работы [7, 8], в которых танки строятся автоматически – с применением методов искусственного интеллекта.

В работе [7] использовано несколько таких методов для управления различными системами танка. Однако их использование не привело к желаемому результату – система управления радаром осуществляла его поворот на 360 градусов, а система передвижения задавала перемещение танка по кругу. В работе [8] используется сравнительно сложный способ представления системы управления в виде особи генетического алгоритма – программы на языке *TableRex*. При этом длина программы составила 7776 бит.

Постановка задачи

Необходимо управлять следующими системами танка: **корпусом, пушкой и радаром**. Управление танком осуществляется следующим образом. В каждый момент времени анализируется текущая игровая позиция (положение танка, его скорость и т.д.) и на основе этой информации формируются четыре действия: передвижение, поворот, поворот пушки, стрельба. Указанные действия вырабатываются в результате интерпретации функции управления, которая генерируется на основе генетических алгоритмов.

Обозначим множество позиций как S, а множество действий танка как A. *Позиция* – элемент некоторого множества всех возможных ситуаций в игре в выбранный момент времени. Позиция включает в себя положение каждого танка, его скорость, направление, угол поворота пушки и радара, энергию, информацию о танке соперника и т. д. Задачу управления танком для рассматриваемой игры сведем к

заданию для каждого момента времени t функции управления $f_t : S \to A$, которая на основании информации о *позиции* определяет действие на объект управления. Упростим поставленную задачу. *Во-первых*, действие танка одинаково во все моменты времени, $f_t = f$. *Во-вторых*, зафиксирован алгоритм управления радаром – вращение по кругу. *В-третьих*, анализируется лишь упрощенная *позиция* – элемент множества $S' = \{< x,\ y,\ dr, tr, w, dh, GH, h, d, e, E >\}$.

Здесь: *x, y* – координаты соперника; *dr* – расстояние, которое осталось «доехать» танку; *tr* – угол, на который осталось повернуться танку; *w* – расстояние от танка до края поля; *dh* – угол между направлением на соперника и пушкой танка; *GH* – угол поворота пушки танка; *h* – направление движения соперника; *d* – расстояние между танком и соперником, *e* – энергия соперника; *E* – энергия нашего танка. *В-четвертых*, множество действий определим как $A' = \{g, p, d, h\}$, где g – угол поворота пушки, p – энергия снаряда, d – расстояние, на которое перемещается танк, h – угол поворота танка.

Итак, задача построения системы управления танком сведена к заданию функции $f : S' \to A'$. Эту функцию будем искать *автоматически* с использованием генетических алгоритмов, а точнее с помощью их разновидности, обладающей высокой эффективностью – программированием с экспрессией генов [9], имеющей преимущества, как перед стандартными генетическими алгоритмами, так и перед генетическим программированием [10].

Представление в виде хромосомы

Для представления в виде особи генетического алгоритма искомой функции управления $f : S' \to A'$ предлагается представлять ее в виде четырех функций f_i, каждая из которых возвращает вещественное число: f_1 – угол поворота пушки, f_2 – энергия снаряда, f_3 – расстояние, на которое перемещается танк, f_4 – угол поворота танка.

В свою очередь, функция f_i представляется в виде *Karva*-дерева – стандартного способа представления для программирования с экспрессией генов. Таким образом, хромосома состоит из четырех строк – линеаризованных *Karva*-деревьев, для представления которых применяются описываемые ниже функции и терминалы.

Требования, предъявляемые к набору базовых функций, используемых при задании хромосомы: набор должен достаточно полным, для того чтобы выразить необходимые зависимости действий от позиции, и не слишком избыточным, для того чтобы работа генетического алгоритма была эффективной. Предлагается использовать следующий набор базовых функций: $+(x, y) = x + y$, $++(x, y, z) = x + y + z$, $n(x) = -x$, $*(x, y) = xy$, $**(x, y, z) = xyz$, min, $s(x) = (1 + e^{-x})^{-1}$, $if>(x, y, z, w) = x > y\ ?\ z : w$.

Набор терминалов строится исходя из определения позиции. Кроме того, этот набор необходимо дополнить некоторым множеством констант. В настоящей работе были выбраны следующие константы: 0, .5, 1, 2, 10.

В табл. 1 приведен пример одной из четырех функций сгенерированной хромосомы и ее запись через базовые функции.

Таблица 1. Пример хромосомы

4 0 7 1 17 4 5 2 4 1 18 9 10 13 12 10 13 22 23 11 17 21 8 15 16 14 21 13 16 18 22 10 16 19 13 8 21 13 10 8 23
f_1 = *(s[n(*(.5, e))], if>[10, *(n(E), dh), *(y, dr, .5), +(w, dr)])

На рис. 1 приведен пример *Karva*-дерева для функции f_1 из табл. 1.

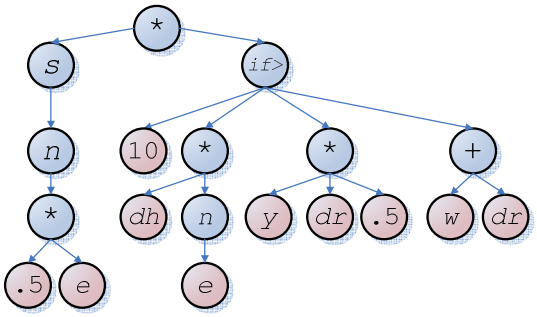

Рис. 1. *Karva*-дерево функции f_1 из табл. 1

Функция приспособленности и критерий остановки процесса эволюции

Функция приспособленности отражает результат поединков против выбранного соперника. При этом отметим, что, так как результат поединка существенно зависит от начального положения танков, то проводится не один поединок, а несколько (в данной работе – двадцать).

Для того чтобы уравнять шансы соперников, выбор начальных положений танков производится следующим образом. Создается случайный набор пар координат $((x_1, y_1), (x_2, y_2))$. После этого для каждой из них проводится два поединка, в первом из которых танк соперника имеет начальные координаты (x_1, y_1), а во втором – (x_2, y_2). Обозначим число очков, набранных нашим танком в бою, состоящим из 20 раундов, как s, а число очков, набранных его соперником, как s_e. При этом приспособленность танка будем вычислять следующим образом: $f = 100 \cdot s / (s + s_e)$.

Процесс эволюции продолжается до тех пор, пока максимальное значение функции приспособленности для элементов популяции не превзойдет некоторого порогового значения (порядка 60-70).

Генетические операции

Для реализации генетического алгоритма в данной работе используются стандартные генетические операции: отбор, мутация и скрещивание. Отбор осуществляется по методу рулетки. Скрещивание хромосом (как наборов из четырех строк) производится поэлементно. При этом для каждой пары строк с вероятностью p в качестве i-го элемента результатом скрещивания выбирается i-й элемент либо первой, либо второй хромосомы, а с вероятностью $(1 - p)$ – скрещивание производится стандартным образом с использованием метода битовой маски. Аналогично скрещиванию, мутация выполняется поэлементно. При этом каждый из символов строки с некоторой вероятностью заменяется некоторым другим. Более подробное описание указанных операций приведено в работе [2].

Эксперименты

При реализации генетического алгоритма число особей в популяции было выбрано равным 50. Один шаг эволюции в популяции такого размера занимает около пяти минут на компьютере *Intel Pentium M* 1.86 *GHz*.

На рис. 2 приведен пример графика, иллюстрирующего работу генетического алгоритма для генерации танка против соперника `sample.Walls`. По оси абсцисс отложен номер поколения, а по оси ординат – значение функции приспособленности.

Рис. 2. Процесс эволюционного построения танка

После генерации каждого танка, для оценки его эффективности были проведены поединки из 100 раундов с тем же соперником, против которого он выводился. В табл. 2 приведены их результаты.

Таблица 2. Результаты поединков (100 раундов)

Соперник	Счет
`newCynic.Cynical`	10933 : 8108
`sample.Walls`	9559 : 7240
`Sample.SpinBot`	11414 : 8556
`sample.TrackFire`	16269 : 9851

В столбце «Счет» первое значение – число очков набранных танком, а второе – его соперником.

Автоматный подход

Эвристическое построение автоматов, управляющих танком, применялось в работе [4]. В тоже время попыток автоматически построить автоматы для управления танком ранее не предпринималось.

В рассмотренном выше подходе, управление танком в любой момент времени осуществляется единственной функцией $f : S' \to A'$. При использовании автоматов эта функция декомпозируется с помощью состояний следующим образом. В каждом состоянии автомата определим функцию $f_i : S' \to A'$, где i – номер состояния.

Для каждой пары различных состояний (i, j) зададим *функцию перехода* $f_{ij} : S' \to R$, построение которой выполняется с помощью генетических алгоритмов, также как это делается для компонент функции управления.

Для управления танком выделим начальное состояние. Далее в каждый момент времени, находясь в состоянии i, последовательно выполняются следующие действия:

- вычисляются значения функций f_{ij} для всех j, $i \neq j$. Как только одно из вычисленных значений оказывается положительным выполняется переход в состояние j;
- определяется значение *функции управления* f_i и выполняется действие, соответствующее найденному значению.

В ходе экспериментов, процесс эволюционного построения танка с автоматной структурой, выводимого в поединках с танком `sample.Walls` был остановлен 36-ом поколении, при значении функции приспособленности равном 93.73. Сгенерированный танк в 1000 раундах побеждает танк соперника со счетом 180168 на 28278 (≈ 86%).

Применение автоматного подхода для описания поведения танка позволяет генерировать более сложные системы управления по сравнению с первым подходом, и, следовательно, создавать более сильные танки.

Заключение

В результате выполнения настоящей работы предложен метод построения системы управления танком для игры *Robocode*. Достоинство этого метода состоит в том, что при его использовании, построение системы управления происходит автоматически. Структуры генерируемых систем управления достаточно просты, но в то же время танки с этими системами управления побеждают любой фиксированный танк из тестового набора. Предложенный метод также позволяет генерировать системы управления с использованием автоматов.

Литература

1. Официальный сайт игры Robocode. http://robocode.sourceforge.net/
2. Mitchell M. An Introduction to Genetic Algorithms. MA: The MIT Press, 1996.
3. Гладков Л. А, Курейчик В. В., Курейчик В. М. Генетические алгоритмы. М.: Физматлит, 2006.
4. Кузнецов Д. В., Шалыто А. А. Система управления танком для игры "Robocode". Вариант 2. СПбГУ ИТМО, 2003. http://is.ifmo.ru/projects/robocode2/
5. Шалыто А. А. SWITCH-технология. Алгоритмизация и программирование задач. СПб.: Наука, 1998. http://is.ifmo.ru/books/switch/1
6. История игры Robocode. http://robowiki.net/cgi-bin/robowiki?History
7. Gade M., Knudsen M., et al. Applying Machine Learning to Robocode. 2003. www.csc.calpoly.edu/~team14fk/F04/dat3_report.pdf

8. Eisenstein J. Evolving robot tank fighters. 2003. http://www.ai.mit.edu/people/jacobe/research/robocode/genetic_tanks.pdf
9. Ferreira C. Gene Expression Programming: A New Adaptive Algorithm for Solving Problems // Complex Systems. 2001. Vol. 13, issue 2, pp. 87–129. www.gene-expression-programming.com/webpapers/GEP.pdf
10. Koza J. R. Genetic programming. On the Programming of Computers by Means of Natural Selection. The MIT Press, MA, 1998.