8. Данилов В.Р. Технология генетического программирования для генерации автоматов управления системами со сложным поведением. – СПбГУ ИТМО, 2006 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/download/danilov_bachelor.pdf, свободный. Яз. рус. (дата обращения 07.02.2011).
9. Царев Ф.Н., Шалыто А.А. Применение генетического программирования для генерации автомата в задаче об «Умном муравье» // Сборник трудов IV-ой Международной научно-практической конференции. – 2007. – Т. 2. – С. 590–597.

Соколов Дмитрий Олегович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, студент, dimoz_88@rambler.ru

УДК 004.85

ПРИМЕНЕНИЕ МАШИННОГО ОБУЧЕНИЯ ДЛЯ СОЗДАНИЯ УПРАВЛЯЮЩИХ АВТОМАТОВ НА ПРИМЕРЕ ИГРЫ «ROBOCODE»

И.И. Чернявский

Рассматривается задача автоматического построения управляющих автоматов. Предлагается метод построения, основанный на применении машинного обучения, а также проводится сравнение предлагаемого метода с методом генетического программирования.

Ключевые слова: машинное обучение, управляющие автоматы, Robocode.

Введение

Построение автономных роботов-агентов является актуальной задачей. Одним из способов описания поведения таких агентов являются управляющие автоматы [1]. Построение автоматов вручную является трудоемким процессом, подверженным ошибкам. В связи с этим внимание исследователей привлек вопрос автоматического создания управляющих автоматов. В настоящей работе этот вопрос рассматривается на примере построения робота-танка для компьютерной игры «Robocode». При этом предлагается метод автоматического построения управляющего автомата для танка и проводится сравнение предлагаемого метода с методом генетического программирования.

Постановка задачи

Задача, решаемая в данной работе, состоит в разработке метода автоматического построения управляющих автоматов. Эта задача рассматривается на примере построения танка для игры «Robocode». Предлагаемый метод должен успешно справляться с задачей построения управляющего автомата для танка, побеждающего заданного противника из поставки игры (танки `sample.Tracker`, `sample.Fire` и `sample.Walls`).

Приведем краткое описание игры «Robocode». Игра представляет собой соревнование роботов-танков на прямоугольном поле. Танк состоит из корпуса, радара и пушки. Программно танк является классом, написанным на языке Java или языках .NET. Этот класс управляет стрельбой, движениями всех частей танка – корпуса, радара и пушки, а также занимается обработкой поступающих событий, к числу которых относятся обнаружение противника радаром, попадание снарядов в танк, поражение других танков, столкновения и т.д.

Игра состоит из последовательности сражений (раундов). В работе рассматриваются только игры с двумя сражающимися танками. В этом случае раунд продолжается до уничтожения одного танка другим. По результатам сыгранных раундов каждому танку начисляются баллы, зависящие от числа выигранных сражений, нанесенного другому танку урона и т.д. Победитель игры определяется наибольшим числом полученных баллов.

В работе [2] рассматриваются задачи «Умный муравей» и «Летающие тарелки», связанные с построением роботов-агентов, управляющих муравьем и летающей тарелкой соответственно, и показывается эффективность генетического программирования [3] в качестве метода построения управляющих автоматов.

В работе [4] исследуется задача автоматического создания управляющих автоматов для роботов-танков в игре «Robocode» при помощи генетического программирования, а в работе [5] предлагается метод двухэтапного генетического программирования для построения автомата.

Обучение с подкреплением

Подход, предлагаемый в настоящей работе, использует методы обучения с подкреплением [6] – класс методов машинного обучения, основной идеей которых является обучение агента через непосредственное взаимодействие с окружающей средой. Общая схема обучения для этого случая приведена на рис. 1.

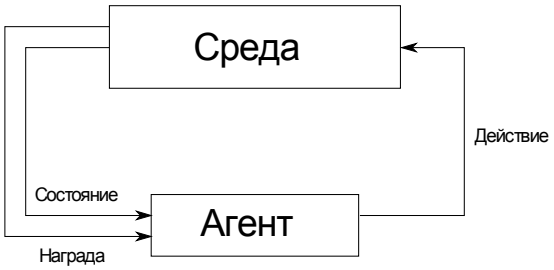

Рис. 1. Схема обучения

Обучение происходит пошагово. На каждом шаге агент «видит» состояние среды, совершает возможное из данного состояния действие и получает ответ от среды – награду. Награда – это число, являющееся оценкой действий агента в ближайшей перспективе. Заметим, что награда может быть отрицательной – быть наказанием. Цель агента – научиться выбирать свои действия так, чтобы максимизировать сумму получаемых им наград. Множества состояний и возможных действий известны заранее и не изменяются в процессе обучения. Также обучаемый агент в общем случае не знает заранее, какую награду дает среда за данное действие из данного состояния. Агент использует для обучения опыт взаимодействия со средой – последовательность из пройденных состояний, совершенных действий и полученных наград.

Рассмотрим метод обучения с подкреплением, используемый в данной работе – *Q*-обучение. При использовании этого метода агент пытается получить как можно более точные оценки значений *Q*-функции: $Q(s, a)$ – ожидание суммы последующих наград, получаемых агентом, находящимся в состоянии s и совершающим действие a. Зная значения этой функции, агент может выбирать правильные действия (из данного состояния достаточно выбрать действие, максимизирующее *Q*-функцию). Начальные оценки значений *Q*-функции полагаются нулевыми.

Пусть на шаге t агент, находясь в состоянии s_t, совершив действие a_t и получив награду r, перешел в состояние s_{t+1}. Тогда оценка $Q(s_t, a_t)$ обновляется по следующей формуле:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(R_t - Q(s_t, a_t)),$$

где R_t – новая оценка:

$$R_t = r_{t+1} + \gamma \max_a Q(s_{t+1}, a).$$

Результатом обучения являются найденные значения *Q*-функции. Полученные значения можно хранить в памяти в виде таблицы. Данный способ представления не всегда удобен – число состояний может быть велико, и поэтому размер таблицы может быть слишком большим. В этом случае для представления *Q*-функции можно использовать нейронную сеть [7]. При таком способе представления для вычисления значения $Q(s, a)$ на входы нейронной сети подаются закодированные состояние s и действие a. Обновление оценки на каждом шаге производится путем применения алгоритма обратного распространения ошибки [7] – входами сети являются закодированные состояние s_t и действие a_t, а желаемым значением на выходе – новая оценка R_t.

Предлагаемый метод

Предлагаемый подход к построению автомата использует идею декомпозиции из работы [5]. Автомат строится в два этапа. На первом этапе строятся состояния автомата, а на втором – функция переходов автомата. В качестве методов, используемых на первом и втором этапах, предлагается использовать методы обучения с подкреплением.

На первом этапе робот обучается выполнению следующих действий:

- преследованию цели;
- уклонению («убеганию») от противника;
- наведению пушки на цель.

Данные действия соответствуют трем состояниям управляющего автомата. Состоянию *A* соответствует преследование цели, *D* – уклонение от противника, *G* – наведение пушки и стрельба. Находясь в каком-либо из состояний, робот выполняет соответствующее состоянию действие.

Цель второго этапа – найти функцию переходов управляющего автомата и, тем самым, получить общую стратегию игры робота, приводящую к победе над противником.

На первом этапе состояния строятся независимо друг от друга. Для построения состояний, как отмечалось выше, используется *Q*-обучение, результатом которого являются значения *Q*-функции. Эта функция может быть представлена в виде таблицы или в виде нейронной сети.

Таким образом, состояние автомата представляется либо в виде таблицы (набор всех значений Q-функции), либо в виде нейронной сети, вычисляющей Q-функцию. Нейронные сети, используемые в данной работе, имеют один скрытый слой. Нейроны этого слоя используют сигмоид-функцию $f(x)=\frac{1}{1+e^{-x}}$. Остальные нейроны используют линейную функцию.

Сформулируем задачи обучения для построения состояний и функции переходов:

1. Обучение преследованию цели
   - состояния среды: два числа – угол между направлением движения и направлением на цель и расстояние до цели;
   - действия: сохранение направления движения или поворот влево/вправо на 10°;
   - награда: «+4» – за приближение к цели.
2. Обучение уклонению от противника
   - состояния среды: относительное положение противника и относительное положение ближайшего препятствия (стены);
   - действия: сохранение направления движения или поворот влево/вправо на 10°;
   - награда: «–5» – за приближение к противнику и «–10» – за столкновение со стеной.
3. Обучение наведению пушки на цель
   - состояния среды: угол между пушкой и целью;
   - действия: поворот пушки влево/вправо на 10° и на 2° или сохранение текущего направления пушки;
   - награда: «+2» – за наведение пушки на цель.
4. Обучение функции переходов
   - состояния среды: номер текущего состояния автомата и расстояние до цели;
   - действия: переход в состояние A, D, G;
   - награда: «+5» – за победу в раунде, «–5» – за поражение (дается на последнем шаге раунда).

Результаты экспериментов

Схема эксперимента. Эксперимент проводится в два этапа. На первом этапе путем решения соответствующих задач обучения с подкреплением строятся состояния автомата. Результатом обучения является таблица или нейронная сеть. Полученный результат можно оценить численно – оценкой является средняя награда, получаемая роботом, использующим сеть, за один шаг.

Для сравнения состояния автомата строятся также с помощью генетического программирования. В данном случае особью является нейронная сеть, вычисляющая Q-функцию. Оценка особей производится путем сравнения среднего значения награды за один шаг.

Результаты, полученные с помощью обучения с подкреплением, сравниваются с результатами, полученными с применением генетического программирования.

На втором этапе эксперимента строится функция переходов автомата. Задача решается методом обучения с подкреплением, а также генетическим программированием – особью в этом случае является тройка нейронных сетей. Находясь в некотором состоянии автомата, робот использует соответствующую данному состоянию нейронную сеть для выбора следующего состояния автомата. Вычисляются выходные значения сети для трех входов, кодирующих возможное следующее состояние и расстояние до цели. Следующее состояние автомата определяется наибольшим полученным значением.

Обучение проводится путем проведения сражений против роботов-противников – `sample.Tracker`, `sample.Fire` и `sample.Walls`. Построенные автоматы сравниваются друг с другом по результатам игры с этими танками.

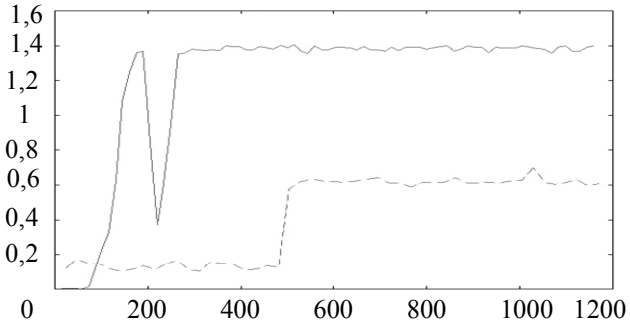

Рис. 2. Графики средней награды в задаче наведения пушки

Наведение пушки. Обучение проводилось с использованием нейронной сети. На рис. 2 сплошной и пунктирной линиями изображены графики средней награды за один шаг при использовании обучения и генетического программирования соответственно. По горизонтали на графиках отложено время, прошедшее с начала эксперимента, по вертикали – средняя награда. Робот обучился наведению пушки: средняя награда 1,35 означает, что на большинстве шагов пушка была направлена на цель. При использовании генетического программирования получено решение со средней наградой 0,7.

Преследование цели. На рис. 3 приведены результаты обучения и генетического программирования. Обучение проводилось с использованием таблиц.

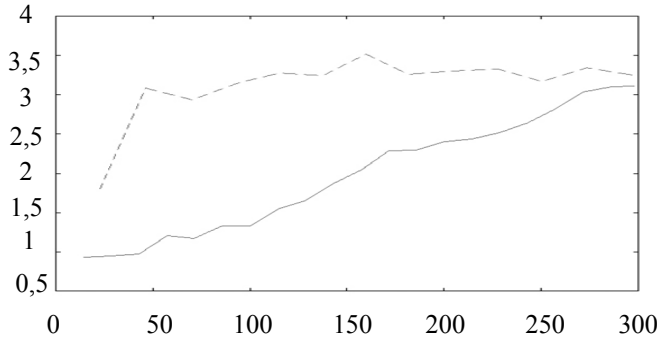

Рис. 3. Графики средней награды в задаче преследования цели

С помощью обучения была достигнута средняя награда за шаг 3,13. Метод генетического программирования быстро находит решение со средней наградой за шаг – 3,51.

Уклонение от противника. Результаты обучения и генетического программирования приведены на рис. 4. Наблюдаются сильные колебания средней награды в процессе обучения. Через 406 секунд была найдена нейронная сеть, соответствующая средней награде, равной «–1,6». Робот, использующий данную сеть, двигается в сторону от противника, почти не сталкиваясь со стенами.

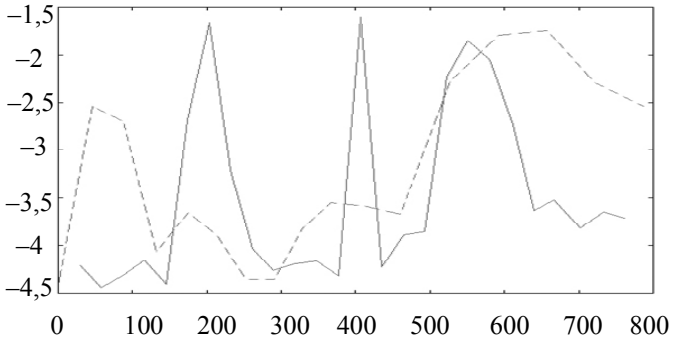

Рис. 4. График средней награды в задаче уклонения от противника

С помощью генетического программирования получено решение со значением средней награды, равным «–1,7».

Построение робота-танка. Роботы `sample.Tracker` и `sample.Fire` являются стандартными роботами поставки и реализуют простые стратегии игры. Робот `sample.Tracker` сначала пытается приблизиться к цели, а затем наводит пушку и стреляет. Робот `sample.Fire` не перемещается по полю, вращает пушку и стреляет из нее при обнаружении цели.

Сравниваемые методы **(**обучение с подкреплением (RL) и генетическое программирование (GP)**)** быстро справляются с построением роботов, побеждающих `sample.Tracker` и `sample.Fire`. С помощью генетического программирования уже на втором поколении строится робот, побеждающий танк `sample.Tracker` с отношением полученных баллов к общей сумме баллов, набранных сражающимися роботами, равным 0,7. Построенный робот наводит пушку, стреляет в цель и не перемещается по полю. Обучение с подкреплением за 500 раундов строит робота, демонстрирующего такую же стратегию игры. Результаты обучения против танков `sample.Tracker` и `sample.Fire` приведены в таблице.

Робот	RL	GP
`sample.Tracker`	8694:21471 (0,71)	9042:21253 (0,70)
`sample.Fire`	7512:22378 (0,75)	3264:18774 (0,85)

Таблица. Результаты обучения

В таблице указаны баллы, полученные роботами после 100 раундов игры, в скобках указано отношение баллов, полученных построенным роботом, к общей сумме баллов.

Робот `sample.Walls` имеет более сложное поведение – двигаясь вдоль стен, робот направляет пушку в сторону поля и стреляет в цель. С помощью генетического программирования было получено решение с отношением баллов (к общей сумме) 0,62. С помощью *Q*-обучения за 500 раундов был построен робот, побеждающий `sample.Walls` с отношением полученных баллов 0,71.

Заключение

В работе рассмотрена задача построения управляющего автомата для робота-агента на примере игры «Robocode». Для решения задачи предложен метод, основанный на двухэтапном построении управляющего автомата с помощью обучения с подкреплением. Предложенный метод успешно справляется с задачей построения автомата, управляющего танком в игре «Robocode».

Сравнение результатов применения *Q*-обучения с результатами, полученными методом генетического программирования, показывает, что на обоих этапах результаты работы методов не сильно отличаются друг от друга. При наведении пушки *Q*-обучение показало лучший результат, в то время как при преследовании цели лучший результат показывает метод генетического программирования. В задаче построения автомата против робота `sample.Fire` лучший результат показало генетическое программирование, а при построении автомата против робота `sample.Walls` – *Q*-обучение.

Полученные данные позволяют утверждать, что методы обучения с подкреплением применимы к построению управляющих автоматов для роботов-танков и не уступают в эффективности генетическому программированию.

Литература

1. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб: Питер, 2009 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/books/_book.pdf, своб.
2. Царев Ф.Н. Разработка метода совместного применения генетического программирования и конечных автоматов. Бакалаврская работа. – СПбГУ ИТМО, 2007 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/papers/_2010_03_03_tsarev.pdf, своб.
3. Mitchell M. An Introduction to Genetic Algorithms. – The MIT Press, 1996.
4. Бедный Ю.Д. Применение генетических алгоритмов для генерации автоматов при построении модели максимального правдоподобия и в задачах управления. Магистерская диссертация. – СПбГУ ИТМО, 2008 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/papers/bednij/, своб.
5. Соколов Д.О. Применение двухэтапного генетического программирования для построения автомата, управляющего моделью танка в игре «Robocode». Бакалаврская работа. – СПбГУ ИТМО, 2009.
6. Sutton R.S., Barto A.G. Reinforcement Learning. An Introduction. – The MIT Press, 1998.
7. Хайкин С. Нейронные сети. Полный курс. – М.: Вильямс, 2006.

Чернявский Илья Игоревич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, студент, chernyavsky@rain.ifmo.ru

УДК 004.4'242

АВТОМАТИЧЕСКИЙ СИНТЕЗ СИСТЕМЫ УПРАВЛЕНИЯ МОБИЛЬНЫМ РОБОТОМ ДЛЯ РЕШЕНИЯ ЗАДАЧИ «КЕГЕЛЬРИНГ»

С.А. Алексеев, А.И. Калиниченко, В.О. Клебан, А.А. Шалыто

Приводится пример автоматического синтеза системы управления мобильным роботом для решения задачи «Кегельринг». Автоматический синтез системы проводится с использованием генетического алгоритма, при помощи которого определяется структура управляющего автомата.

Ключевые слова: автоматное программирование, генетические алгоритмы, автоматический синтез систем управления.

Введение

Для построения систем управления мобильными роботами целесообразно использовать технологию автоматного программирования [1–3], в которой, в частности, предлагается строить программу как систему автоматизированных объектов управления, в которых управляющая программа представляет собой систему автоматов, взаимодействующих между собой за счет вложенности и вызываемости. Использование автоматного подхода при создании подобных систем обладает рядом достоинств, таких как возможность повышения уровня автоматизации процесса верификации, документируемость, упрощение внесения изменений и т.д.