9. Айфичер Э., Джервис Б. Цифровая обработка сигналов: практический подход. – М.: Вильямс, 2008.
10. Sharp distance sensors [Электронный ресурс]. – Режим доступа: http://www.parallax.com/dl/docs/prod/acc/SharpGP2D12Snrs.pdf, свободный. Яз. англ. (дата обращения 09.02.2011).
11. Туккель Н.И., Шалыто А.А. Система управления дизель-генератором (фрагмент). Программирование с явным выделением состояний. Проектная документация [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/projects/dg/, свободный. Яз. рус. (дата обращения 09.02.2011).
12. Туккель Н.И., Шалыто А.А. От тьюрингова программирования к автоматному // Мир ПК. – 2002. № 2. – С. 144–149 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/works/turing/, свободный. Яз. рус. (дата обращения 09.02.2011).
13. Шалыто А.А. Switch-технология. Алгоритмизация и программирование задач логического управления. – СПб: Наука, 1998. – 628 с.
14. Шалыто А.А. Логическое управление. Методы аппаратной и программной реализации. – СПб: Наука, 2000. – 780 с.
15. Шопырин Д.Г., Шалыто А.А. Синхронное программирование // Информационно-управляющие системы. – 2004. – № 3. – С. 35–42 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/works/sync_prog/, свободный. Яз. рус. (дата обращения 09.02.2011).
16. Клебан В.О., Шалыто А. А. Использование автоматного программирования для построения многоуровневых систем управления мобильными роботами // Сборник тезисов 19 Всероссийской научно-технической конференции «Экстремальная робототехника». – СПб: ЦНИИ РТК, 2008. – С. 85–87.
17. Brooks R.A. A Robust Layered Control System for a Mobile Robot // IEEE Journal of Robotics and Automation. – 1986. – V. 2. – P. 14–23.
18. Harel D., Pnueli A. On the development of reactive systems / In «Logic and Models of Concurrent Systems». – NATO Advanced Study Institute on Logic and Models for Verification and Specification of Concurrent Systems. – Springer Verlag, 1985. – P. 477–498.
19. Flying Machine Arena [Электронный ресурс]. – Режим доступа: http://www.idsc.ethz.ch/Research_DAndrea/FMA, свободный. Яз. англ. (дата обращения 09.02.2011).

Клебан Виталий Олегович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, аспирант, vk.developer@gmail.com

Шалыто Анатолий Абрамович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, доктор технических наук, профессор, зав. кафедрой, shalyto@mail.ifmo.ru

УДК 004.832.28

ПРИМЕНЕНИЕ ДВУХЭТАПНОГО ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ ДЛЯ ПОСТРОЕНИЯ МОДЕЛИ ТАНКА В ИГРЕ «ROBOCODE»

Д.О. Соколов

Рассматривается применение генетического программирования для построения конечных автоматов, управляющих системами со сложным поведением. Приведен метод двухэтапного генетического алгоритма, основанный на идеях динамического программирования. Рассмотрено применение этого метода на примере игры «Robocode».
Ключевые слова: автомат, деревья разбора, генетическое программирование.

Введение

Задача построения управляющих систем для беспилотной техники в настоящее время является актуальной. Чаще всего эта задача решается вручную. Такой подход не всегда является эффективным как ввиду больших затрат ресурсов, так и в связи с низким качеством построенных систем. В некоторых случаях ввиду сложности системы построение управляющей системы при помощи ручного труда невозможно. Естественная идея – поручить построение управляющей системы компьютеру.

Эволюционные вычисления успешно применяются для автоматического создания программ [1]. Эффективность применения эволюционных алгоритмов напрямую зависит от способа представления программы в виде хромосомы [2]. Для многих управляющих систем их поведение удобно представлять в виде конечных автоматов [3, 4]. Одной из таких систем является система управления танком в игре «Robocode», которая и рассматривается в настоящей работе.

Выделим основные проблемы, которые решаются в данной работе, возникающие при использовании генетического программирования для решения задачи управления:

- большое время работы генетического алгоритма;
- необходимость обработки вещественных переменных при помощи конечных автоматов;
- попадание в локальные максимумы.

Автор предполагает, что читатель знаком с основными понятиями генетических алгоритов: особь, генетические операторы (скрещивание, мутация), фитнесс-функция, поколение, классический генетический алгоритм.

Постановка задачи управления

Для описания предлагаемого метода была выбрана задача управления танком в игре «Robocode». У танка необходимо управлять следующими устройствами: радаром (*Radar*), пушкой (*Gun*) и системой движения (*Body*).

Управление танком осуществляется аналогично работе [5]. В каждый момент времени анализируется текущая ситуация (положение танка, его скорость и т.д.). На основе этой информации формируются четыре действия: передвижение (вперед/назад), поворот, поворот пушки, стрельба. Указанные действия формируются в результате интерпретации функции управления, которая генерируется на основе генетических алгоритмов.

Конкретизируем постановку задачи управления. Обозначим текущее время в игре как $t \in N$. 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.

Задача управления для рассматриваемой игры в общем случае состоит в задании для каждого момента времени t функции управления $f_t : S^t \Rightarrow A$, которая на основании информации о позициях во все предыдущие моменты времени (получаемой из среды «Robocode») определяет действие на объект управления в текущий момент времени. Таким образом, решение задачи управления танком – вектор $[f_1, f_2, \ldots]$.

Упростим задачу управления. Пусть:

1. действие танка в текущий момент времени зависит не только от позиции в этот момент времени, а также от состояния самой системы управления;
2. зафиксирован алгоритм управления радаром – вращение по кругу;
3. при выборе действия анализируется лишь упрощенная позиция – элемент множества $S' = (x, y, dr, tr, w, dh, GH, h, d, e, E)$, здесь: *x*, *y* – координаты танка соперника относительно нашего танка; *dr* – расстояние, которое осталось «доехать» нашему танку (после вызова метода *AdvancedRobot.setAhead*); *tr* – угол, на который осталось повернуться нашему танку; *w* – расстояние от нашего танка до края поля; *dh* – угол между направлением на танк соперника и пушкой нашего танка; *GH* – угол поворота пушки нашего танка; *h* – направление движения танка соперника; *d* – расстояние между нашим танком и танком соперника; *e* – энергия танка соперника; *E* – энергия нашего танка;
4. множество действий $A' = g, p, d, h$, где *g* – угол поворота пушки; *p* – энергия снаряда (неположительные значения означают, что выстрел не производится); *d* – расстояние, на которое перемещается танк; *h* – угол поворота танка.

Учитывая изложенное, задача построения системы управления танком сведена к заданию функции $f : S' \times S \to A' \times S$, где *S* – конечное множество состояний системы. Естественно представлять данную функцию в виде автомата, который будем искать при помощи генетического алгоритма.

Основные идеи

Как отмечалось во введении, в данной работе рассматривается метод решения трех проблем, возникающих при использовании генетических алгоритмов для решения задачи управления. Рассмотрим методы решения.

Попадание в локальные максимумы. Для решения данной проблемы использован островной генетический алгоритм [6] (рис. 1), а также оператор большой мутации.

Основные отличия островного алгоритма от классического генетического:

– разделение популяции на несколько популяций, развивающихся независимо – разделение на острова;
– добавление оператора миграции [6].

Оператор большой мутации на фиксированной доле островов заменяет поколение на случайное. Данный оператор применяется через фиксированное число поколений.

Обработка вещественных переменных. Как уже отмечалось, будем строить конечный детерминированный автомат с множеством состояний *S*. В каждое состояние поместим обработчик переменных. Обработчик *i*-ого состояния представляет собой функцию $f_i : S' \to A'$ (см. раздел «Постановка задачи управления»).

Зафиксируем некоторое число k, $2^k \le |S|$, сопоставим автомату еще один обработчик переменных – $g : S' \to \{0,1\}^k$. Функция переходов автомата будет иметь вид $G : S \times S' \to S, G(s) = H(S, g(S'))$.

Задача получения автомата свелась к задаче получения функций f_i, g и H. При помощи выделения обработчиков, обработка вещественных переменных была «отделена» от автомата, таким образом, появилась возможность применять любой из имеющихся методов для получения функции H.

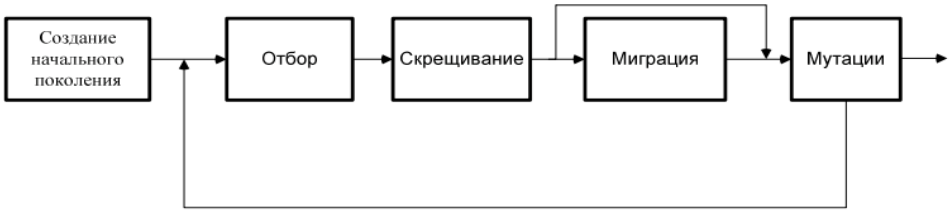

Рис. 1. Схема островного генетического алгоритма

Уменьшение времени работы генетического алгоритма. Время работы напрямую связано с размером пространства решений. Для уменьшения размера данного пространства будем использовать идеи динамического программирования [7].

В результате применения метода деревьев разбора задача разбилась на две части:

1. получение f_i;
2. получение g и H.

В соответствии с принципом динамического программирования необходимо сначала решить подзадачи оптимальным образом. Подзадачами будем считать нахождение f_i. В нашем случае сложно сформулировать, что значит «оптимально», поэтому на первом этапе при помощи генетического алгоритма сформируем некоторое множество f_i – репозиторий обработчиков (состояний). На втором этапе будем получать функции g и H, но вместо случайных f_i будем случайным образом выбирать их из репозитория (сразу отметим, что финальная реализация будет несколько отличаться от этого, по причине того, что можно развить эту идею, пользуясь особенностями задачи).

Таким образом, осталось разработать метод генерации вещественных функций.

Представление обработчика вещественных переменных

Предлагаемый метод представления особи является комбинацией методов, описанных в работах [5, 8], f_i (см. определение в разд. «Основные идеи») будет представлять собой четверку так называемого дерева разбора.

Дерево разбора представляет собой дерево, в котором во внутренних вершинах находятся функции. У каждой внутренней вершины ровно столько потомков, какова арность функции в данной вершине. В листья подаются входные переменные или заранее зафиксированные константы.

В данной работе, аналогично работе [5], использованы внутренние функции $if\,(x < y)\ return\ w\ else\ return\ v$; $\min(x, y)$; $x \times y$; $-x$; $\dfrac{1}{1+\exp(-x)}$; $x + y + w$; $x \times y \times w$ и константы: 0,1; 0,5; 1; 2; 5; 10.

Пример дерева разбора изображен на рис. 2.

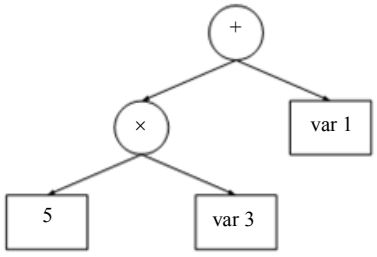

Рис. 2. Пример дерева разбора

Условное обозначение var i означает, что в данных лист подается значение i-ой переменной. Генетические операторы для деревьев разбора аналогичны описанным в работе [8]. Таким образом, ав-

томат представляет собой массив, состоящий из четверок деревьев разбора, k деревьев разбора – функция g, а также таблица переходов – функция H.

Генетический алгоритм

Как говорилось ранее, на всех этапах построения используется островной генетический алгоритм. На первом этапе строятся четверки деревьев разбора, эти четверки сохраняются в репозитории. Причем для каждого из деревьев известно, за какое из действий танка он отвечает. На втором этапе генерируется функция переходов для автомата.

Общие для двух этапов элементы генетического алгоритма

В качестве основной стратегии формирования следующего поколения используется элитизм. При рассмотрении текущего поколения отбрасываются все особи, кроме некоторой доли наиболее приспособленных – элиты. Эти особи переходят в следующее поколение. После этого оно дополняется в определенной пропорции случайными особями – особями из текущего поколения, которые мутировали и являются результатами скрещивания особей из текущего поколения (отдельно отметим, что скрещиваться могут не только элитные особи, а все). Особи, «имеющие право» давать потомство, определяются «в поединке»: выбираются две случайные пары особей, и более приспособленная особь в каждой из них становится одним из родителей. Через фиксированное число поколений каждый остров меняется с другим случайным числом случайно выбранных элитных особей.

Через заранее заданное число поколений происходит *большая мутация* – фиксированная доля островов заменяется островами со случайными особями. Проведение мутации в момент, когда функция приспособленности «элитных» особей изменяется незначительно, невозможно, так как за счет миграции особей между островами среднее значение функции приспособленности постоянно изменяет свое значение.

Первый этап генетического алгоритма

На первом этапе генетического алгоритма выращиваются четверки деревьев разбора. Начиная с заранее заданного поколения, лучшая особь поколения добавляется в репозиторий. При генерации начального поколения все острова заполняются случайно сгенерированными особями. Скрещивание деревьев разбора происходит попарно.

Для подсчета функции приспособленности создается автомат из одного состояния (для него не нужны функции g и H), для которого f_1 является тестируемой четверкой деревьев разбора. При подсчете фитнесс-функции используется моделирование соревнования между танком, управляемым тестируемой особью, и выбранным танком. Заранее выбрать фитнесс-функцию не представляется возможным, так как на первом этапе генерируется не вся особь в целом, а лишь одно состояние. В данной работе для сравнения использовались различные фитнесс-функции:

- $t.damage + t.bulletdamage$;
- $\dfrac{(t.survival - n)n}{t.bulletDamage}$;
- $\dfrac{t.score + t.survival * t.bulletDamage}{t.score + t.survival * t.bulletDamage + e.score + e.survival * t.bulletDamage}$;
- $\dfrac{t.score}{t.score + e.score}$;
- $t.score$.

Здесь: t – модель танка, управляемая тестируемой особью; e – модель танка противника; *damage* – ущерб, нанесенный противнику за счет попаданий и столкновений; *bulletDamage* – ущерб, нанесенный противнику только за счет попаданий; *survival* – число раундов, в которых танк выжил; *score* – число очков, набранных танком (вычисляется средой «Robocode»); n – число раундов в соревновании. Для избежания излишней пассивности танков (отсутствия стрельбы) часть запусков алгоритма проводилась с запретом на движение тестируемого танка.

Второй этап генерации особей

После первого этапа становится доступен репозиторий с деревьями разбора, где деревья сгруппированы по действиям, за которые они отвечают.

Теперь особь представляет собой тройку (S, H, g), где S – множество состояний, каждое из которых представляет собой четверку деревьев разбора из репозитория; H – функция переходов автомата, представляет собой полную таблицу переходов, генетические операторы аналогичны операторам, описанным в работе [9]; g – подфункция H, по состоянию в игре возвращающая номер перехода, представляет собой k деревьев разбора.

При скрещивании элементы данной тройки скрещиваются попарно. При скрещивании S и S' элементы этих множеств также скрещиваются попарно. При скрещивании состояний P и Q возможны следующие ситуации (наследники P' и Q'):

– $P' := P, Q' := Q$;
– $P' := Q; Q' := P$;
– P' и Q' случайным образом выбирают деревья разбора от родителей.

При генерации случайного автомата, для каждого состояния из репозитория равновероятно выбирается по одному дереву разбора, отвечающему за каждое действие (в этом отличие от раздела «Основные идеи»: функции f_i можно разбить на части). Стартовое состояние генерируется случайным образом. Деревья разбора, отвечающие за функцию g, генерируются случайным образом.

Также отметим, что операторы скрещивания и мутации к деревьям разбора, находящимся внутри состояний, не применяются.

На этом этапе использовалась следующая фитнесс-функция: $\frac{t.score}{t.score + e.score}$.

Основные особенности применения двухэтапного алгоритма

Для начала оценим размер пространства решений в случае применения двухэтапного генетического алгоритма.

Для деревьев, отвечающих за логические переменные, соотношение не изменилось: $(Z^E)^{V_c}$. Также не изменилось соотношение переходов: $N^{2^{V_c}}$. Однако для генерации состояния теперь достаточно лишь выбрать четыре дерева из репозитория. В этом случае имеет место соотношение S^4, где S – размер репозитория. Таким образом, $|\Omega_{tree(new)}| = (Z^E)^{V_c} S^4 N^{2^{V_c}}$. При этом $|\Omega_{tree(new)}| << |\Omega_{tree}|$.

Основные настройки генетического алгоритма

Перечислим настройки генетического алгоритма, используемые при генерации особей: число островов – 4; число особей на острове – 52; доля элитных особей – 0,15; вероятность мутации – 0,02; период миграции – 11; период большой мутации – 140; доля островов, уничтожаемых при большой мутации – 0,8; k=2.

Результаты

В качестве основных соперников использовались следующие танки из стандартного набора игры «Robocode»: `sample.Fire; sample.Tracker; sample.Target; sample.Walls`.

На рис. 3 представлен граф переходов автомата, построенного с помощью генетического алгоритма.

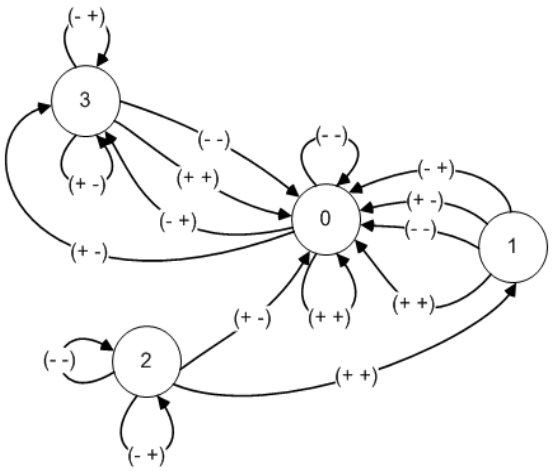

Рис. 3. Граф переходов автомата, полученного в результате работы второго этапа генетического алгоритма

Поясним используемые обозначения. На ребрах графа переходов находятся отметки вида (+, –). Они указывают на значение i-го бита функции g (+ соответствует единице).

Структуру состояний изобразить не представляется возможным из-за размеров деревьев разбора.

Результаты соревнования данной особи проиллюстрированы на рис. 4.

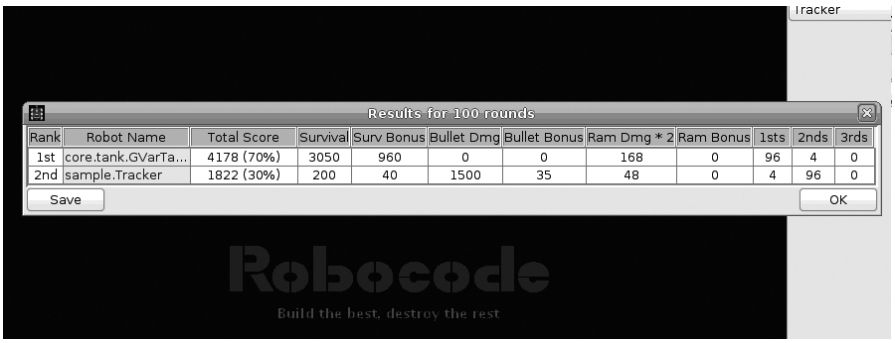

Рис. 4. Результаты соревнования

На первом этапе проводились запуски танка для всех вариантов фитнесс-функции против всех соперников. Таким образом, размер репозитория составил 304 дерева разбора для каждого из четырех типов.

По результатам сравнения предлагаемый в настоящей работе метод превосходит методы, предложенные в работе [5]. При этом отметим, что разница в результатах при применении методов с применением конечных автоматов и *karva*-деревьев лежит в пределах погрешности измерений функции приспособленности. Однако при использовании метода, предложенного в данной работе (несмотря на вдвое увеличенное число раундов соревнований при подсчете фитнесс-функции), наблюдается по сравнению с работой [5] ускорение процесса получения особи (на втором этапе) с заданным значением функции приспособленности в среднем в полтора раза (примерно 1,5–2 часа).

Также данный метод позволяет достичь большей универсальности, так как на первом этапе генерируются различные *хорошие* стратегии поведения – каждое состояние представляет собой уже законченную стратегию поведения против конкретного поведения противника.

Из недостатков предложенного метода отметим длительное время проведения первого этапа алгоритма. На создание репозитория из 300 четверок деревьев потребовалось восемь суток.

Заключение

В данной работе задача построения управляющего автомата для систем со сложным поведением разбита на части:

1. Построение обработчиков вещественных переменных в одном отдельно взятом состоянии. При этом возможна более точная настройка желаемого поведения в данном состоянии;
2. Построение графа переходов и вспомогательных конструкций (например, дополнительные деревья разбора для генерации булевых переменных).

Разбиение на части позволило улучшить качество решений и сократить требуемые вычислительные ресурсы (для второго этапа) в случае, когда подсчет для первого этапа выполнен заранее. К недостаткам метода можно отнести длительное время работы первой части алгоритма.

Для каждой части были реализованы модификации островного генетического алгоритма.

Литература

1. Holland J.P. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. – The University of Michigan Press, 1975.
2. Koza J.R. Genetic programming: On the Programming of Computers by Means of Natural Selection. – Cambridge: MIT Press, 1992.
3. Aho A., Sethi R., Ullman J. Compiler Design: Principles, Tools, and Techniques. Наука, MA: Addison Wesley, 1986.
4. Шалыто А.А. Switch-технология. Алгоритмизация и программирование задач логического управления. – СПб: Наука, 1998. – 628 с.
5. Бедный Ю.Д. Применение генетических алгоритмов для генерации автоматов при построении модели максимального правдоподобия и в задачах управления. – СПбГУ ИТМО, 2006 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/papers/bednij/masters.pdf, свободный. Яз. рус. (дата обращения 07.02.2011).
6. Яминов Б. Генетические алгоритмы. – СПбГУ ИТМО, 2005 [Электронный ресурс]. – Режим доступа: http://rain.ifmo.ru/cat/view.php/theory/unsorted/genetic-2005/, свободный. Яз. рус. (дата обращения 07.02.2011).
7. Cormen T., Leiserson C., Rivest R., Stein C. Introduction to Algorithms. 3rd edition. – Cambridge: MIT Press, 2009.

.

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.