УДК 004.89

АЛГОРИТМЫ УПРАВЛЕНИЯ ИНТЕЛЛЕКТУАЛЬНЫМИ АГЕНТАМИ В КОМПЬЮТЕРНЫХ ИГРАХ

А.В. Экало
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»
Россия, 197376, Санкт-Петербург, ул. Профессора Попова, дом 5
E-mail: ekalo@nicetu.spb.ru

С.А. Беляев
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»
Россия, 197376, Санкт-Петербург, ул. Профессора Попова, дом 5
E-mail: bserge@bk.ru

Ключевые слова: компьютерные игры, машинное обучение, интеллектуальный агент, алгоритмы управления, мультиагентные системы.

Аннотация: Авторы рассматривают актуальную задачу управления в компьютерных играх. Приведены общие описания основных алгоритмов управления интеллектуальными агентами, как основанные на машинном обучении, так и традиционные подходы. Для каждого алгоритма перечислены возможные варианты использования в играх. Авторы перечисляют несколько платформ проведения экспериментов для сравнения эффективности применения разработанных алгоритмов.

1. Введение

Современные компьютерные игры являются платформой для отладки новейших алгоритмов управления. Теория игр изучает оптимальные стратегии в играх, на её основе разработаны решения для игры против человека в шашки, шахматы, го. В случае шашек разработана стратегия, при правильном исполнении которой обоими противниками всегда будет ничья, а при отклонении одного из противников от стратегии он проиграет [1].

В качестве платформы для изучения универсальных алгоритмов долгое время выступала Atari 2600. Особенностью её реализации является то, что в ней отсутствует датчик случайных чисел, а значит при одной и той же последовательности действий игрока получается один и тот же результат. Тем не менее данная платформа позволяет исследовать большое количество алгоритмов машинного обучения.

Не менее популярной для изучения мультиагентных систем является виртуальный футбол. На базе современной реализации [2] не только исследуются вопросы кооперации, совместного решения задач и взаимодействия интеллектуальных агентов, но и проводятся ежегодные соревнования

В рамках данной работы рассматриваются основные подходы и алгоритмы, используемые в современных компьютерных играх в части задач противодействия игроку-человеку. В качестве объекта настоящего исследования выступает интеллектуальный агент, противодействующий человеку или другому агенту, в качестве предмета –

подходы и алгоритмы управления интеллектуальным агентом в играх с нулевой суммой. Вопросы создания контента игр и моделирования игрока выходят за рамки исследования.

2. Подходы и алгоритмы управления

2.1. Традиционные подходы

Конечные автоматы [3] и их вариации в виде иерархических конечных автоматов (hierarchical finite state machines – HFSM) активно используются при программировании поведения интеллектуальных агентов. При этом состояния автомата представляют собой решаемую задачу, переходы обеспечивают проверку условий и переход в новое состояние, также задаётся набор действий, который должен выполняться в каждом состоянии. Они просты в описании и программировании, но к ним не применимы подходы по обучению.

Поведенческие деревья (behavior trees) [4] представляют собой экспертное решение задачи задания поведения интеллектуального агента и описываются по аналогии с конечными автоматами. Они позволяют обеспечить последовательное выполнение действий в дочерних узлах, обеспечить вероятностный выбор дочерних узлов либо выбор на основании приоритета, задать количество повторов действий в дочерних узлах или ограничения по времени. Поведенческие деревья представляют собой наглядную организационную структуру, но при задании сложных условий и описании нестандартных шаблонов поведения могут оказаться трудоёмкими в реализации и поддержке.

Разработка поведения интеллектуального агента может строиться на основе функции полезности (utility-based) [5]. Функция полезности позволяет сравнить несколько возможных исходов. Выбор наиболее подходящего обычно осуществляется по принципу максимизации значения функции полезности нового состояния. Ограничением применения данного подхода являются сложности при описании функции полезности для сложных состояний, тем не менее даже приблизительная функция полезности может выступать в качестве инструмента для ускорения работы алгоритмов неинформированного поиска. В качестве алгоритмов неинформированного поиска в графе традиционно рассматривают поиск в глубину (depth-first search – DFS) и поиск в ширину (breadth-first search – BFS) [6]. Данные алгоритмы редко используются в играх, за исключением итеративного поиска в ширину, когда поиск в ширину ограничивается заданным значением глубины с её пошаговым увеличением.

Одним из наиболее известных алгоритмов поиска по первому наилучшему совпадению (best-first search) является A* [1, 6]. Алгоритм осуществляет поиск пути во взвешенном графе и основывается на использовании эвристической функции, которая вычисляет оценку длины пути от заданного узла до целевого узла в графе. Алгоритм вычисляет расстояние до всех связанных с исходным узлом узлов, помещает их в список открытых узлов, затем для всех узлов в списке открытых узлов использует эвристическую функцию для оценки расстояния до цели. Процедура повторяется для узла с минимальной суммой дистанции и оценки расстояния для цели. Алгоритм эффективно работает на графах, в которых существует большое количество маршрутов из одного узла в другой, например, для графов, описывающих движение по открытой местности, использованием евклидового расстояния в качестве эвристической функции. Существуют вариации алгоритма для динамически изменяющейся окружающей среды – D*.

При принятии решения интеллектуальным агентом может использоваться алгоритм минимакса (minimax) [1, 6]. Данный алгоритм предполагает намерение игрока максимизировать, а его оппонента – минимизировать получаемый результат. Оценка состоя-

ния при этом может осуществляться, например, с использованием функции полезности. Минимакс разработан в рамках теории игр и эффективно работает при применении к деревьям состояний с невысоким фактором ветвления. Для повышения эффективности алгоритма применяются его модификации в виде альфа-бета отсечений [6], которые позволяют не рассматривать все ветви дерева поиска.

При работе с большими деревьями состояний, такими как в шахматах и го, применение минимакса даже с применением альфа-бета отсечений требует слишком много времени. В таких случаях может применяться алгоритм поиска по деревьям Монте Карло (Monte Carlo tree search – MCTS) [7]. Алгоритм в соответствии с заданной стратегией выбирает узел из дочерних узлов, для него выполняется связанное действие, затем с использованием случайного выбора дочерних узлов осуществляется движение по дереву вплоть до окончания игры. Полученный результат (победа, поражение или ничья) распространяется в обратном направлении. Алгоритм выполняется многократно, что позволяет оценить вероятность победы в зависимости от выбранного узла. MCTS использовался в программе AlphaGo, которая обыграла профессионалов игры в го [1]. У данного алгоритма есть важное ограничение – он применим только в случае, если есть возможность симулировать действия при движении по дереву состояний и оценивать результат.

2.2. Оптимизация параметров

При выборе действия интеллектуальным агентом может выполняться оптимизация заданных параметров, например, значений функции полезности. Для решения данной задачи может использоваться локальный поиск, один из простейших алгоритмов – поиск восхождением к вершине (hill climbing) [8]. Случайным образом в пространстве решений выбирается одно решение, определяются все «соседние» решения, вычисляется функция соответствия для всех найденных решений, если ни одно из соседних решений не лучше первого, то поиск завершается, иначе выбирается лучшее и поиск осуществляется относительно него. Существуют модификации алгоритма, когда выбор осуществляется с использованием градиента (gradient-based hill climber) или с использованием мутаций (randomized hill climber). Более совершенный алгоритм оптимизации параметра – имитация отжига (simulated annealing). Все перечисленные алгоритмы находят только локальный минимум.

Для поиска могут использоваться и эволюционные алгоритмы (evolutionary algorithm), такие как генетические алгоритмы (genetic algorithm – GA) [1]. В основе генетического алгоритма находятся популяции, которые на первом шаге формируются случайным образом [6]. Каждый член популяции (индивидуум) оценивается с использованием функции пригодности. Для перехода к следующему шагу выполняется скрещивание индивидуумов, после скрещиваний с использованием функции пригодности выбираются индивидуумы, которые перейдут в следующую популяцию. В процессе скрещивания с небольшой вероятностью могут возникать мутации. Использование алгоритмов ограничивается эффективностью используемой функции пригодности.

2.3. Использование обучения

Обучение с учителем подразумевает наличие большого количества примеров, в каждом из которых для заданного набора входных параметров указан ожидаемый результат. На этапе обучения алгоритму предъявляется пример и алгоритм обучается генерировать ожидаемый результат с минимальной ошибкой по всем примерам. Одной из наиболее известных структур, использующих такой подход, являются искусственные нейронные сети (artificial neural network – ANN) [6,8]. Существует большое количество типов нейронных сетей, но все они строятся на основании персептрона, который имеет

взвешенные входные параметры, они суммируются и подаются на вход функции активации, а она в свою очередь определяет – активировался персептрон или нет. Множество персептронов соединяются друг с другом и образуют нейронную сеть, в которой результаты активации одних персептронов передаются на вход другим персептронам и так далее. Главное назначение такой сети – аппроксимация функции, которая по заданным входным параметрам генерирует требуемые выходные параметры. Если функция известна заранее, то ANN не нужна. Могут использоваться разные подходы для обучения нейронной сети, чаще всего используют метод обратного распространения ошибки: на вход ANN подаются входные параметры, в соответствии со связями между персептронами по сети передаются сигналы, как результаты активации, на выходе сети получается её результат, результат сравнивается с ожидаемым, вычисляется ошибка, которая распространяется в обратном порядке с соответствующим изменением весов по всей сети. Примеры предъявляются ANN многократно для минимизации ошибки по всем примерам. Главное преимущество нейронной сети – возможность игнорировать шум во входных параметрах, главный недостаток – продолжительное обучение. Существуют различные подходы к сокращению времени обучения [9]. При управлении в играх ANN могут применяться как для распознавания образов, так и для принятия решений.

Метод опорных векторов (support vector machine) [10] активно используется для задач классификации и регрессионного анализа. При использовании данного метода входные параметры рассматриваются как вектора, они переводятся в пространство более высокой размерности, затем выполняется поиск гиперплоскости, разделяющей их с максимальным зазором. Главные преимущества метода опорных векторов в том, что они находят глобальное решение и при использовании данного метода проще решать проблему переобучения.

Деревья решений (decision tree) позволяют реализовывать простые и эффективные алгоритмы управления интеллектуальными агентами. В узлах дерева выполняется проверка входных параметров и осуществляется переход по соответствующим ветвям вниз, к следующим узлам, в листьях указывают действия. Разработаны эффективные методы обучения деревьев решений [11]. В играх чаще всего используются алгоритмы ID3 и C4.5. Основная идея ID3: набор входных параметров анализируется с целью выбора параметра, деление по которому даёт максимум информации, формируются соответствующие дополнительные узлы дерева, затем деление повторяется для вновь созданных узлов. К недостаткам деревьев решений можно отнести сложности в работе с зашумлёнными параметрами и громоздкостью автоматически построенных деревьев.

N-граммы (n-grams) [8, 12] используются для предсказания возникновения одинаковых последовательностей из N элементов. При анализе обучающей последовательности элементов рассматриваются все одинаковые подпоследовательности из N-1 элемента и для каждой вычисляются возможные N-е элементы и их вероятности. N-граммы имеют широчайшее применение от проверки текстов на плагиат до предсказания следующего действия пользователя в игре.

Обучение с подкреплением (reinforcement learning – RL) [13,14] применяются для формирования и использования положительной или отрицательной обратной связи для заданной последовательности действий. Положительная обратная связь предполагает, что действия выполняются правильно, отрицательная обратная связь говорит о необходимости отказа от данной последовательности действий. Алгоритм обучается выполнению такой последовательности действий, которая максимизирует суммарное вознаграждение. Одним из наиболее распространённых является Q-обучение (Q-learning), в котором вознаграждение формируется на основании функции полезности Q.

3. Платформы для проведения экспериментов

Для обеспечения сравнимости результатов и в некоторых случаях для проведения соревнований предлагается использовать общедоступные платформы для проведения экспериментов:

- Arcade Learning Environment (ALE) используется для экспериментов с Atari 2600 (https://github.com/mgbellemare/Arcade-Learning-Environment);
- OpenAI Gym используется для разработки и сравнения алгоритмов обучения с подкреплением (https://gym.openai.com);
- The General Video Game AI Competition используется для сравнения универсальных алгоритмов управления (http://www.gvgai.net);
- VIZDoom используется для сравнения алгоритмов управления на основе видеоизображения (http://vizdoom.cs.put.edu.pl);
- Angry Birds AI Competition для сравнения алгоритмов управления в вероятностных играх (http://aibirds.org);
- The RoboCup Soccer Simulator используется для разработки алгоритмов управления в реалистичных мультиагенных системах (https://github.com/rcsoccersim).

4. Заключение

Все рассмотренные в исследовании подходы и алгоритмы используются при создании интеллектуальных агентов в компьютерных играх, апробируются и сравниваются в приведённых платформах для проведения экспериментов, имеют широкое практическое применение. Использование подходов искусственного интеллекта не является панацеей [8], поэтому перед их использованием следует оценить эффект от применения.

Список литературы

1. Yannakakis G., Togelius J. Artificial intelligence and games. Springer, 2018. 350 p.
2. The RoboCup Soccer Simulator. https://github.com/rcsoccersim
3. Гилл А. Введение в теорию конечных автоматов. М., 1966, 272 с.
4. Champandard A.J.. Behavior trees for next-gen game AI (Video, Part 1). http://aigamedev.com/open/article/behavior-trees-part1/
5. Mark D. Behavioral Mathematics for game AI. Charles River Media, 2009. 480 p.
6. Рассел С., Норвиг П. Искусственный интеллект: современный подход / 2-е изд. М.: Вильямс, 2016. 1408 с.
7. Jacobsen E.J., Greve R., Togelius J. Monte Mario: platforming with MCTS // Annual Conference on Genetic and Evolutionary Computation (GECCO '14). ACM. New York, 2014. P. 293-300.
8. Millington I., Funge, J. Artificial intelligence for games / 2nd ed. Burlington: Elsevier. 2009. 896 p.
9. Коробов Д.А., Беляев С.А. Современные подходы к обучению интеллектуальных агентов в среде Atari // Программные продукты и системы. 2018. Т. 31, № 2. С. 284-290.
10. Вьюгин В. Математические основы теории машинного обучения и прогнозирования. М.: МЦМНО, 2013. 390 с.
11. Паклин Н.Б., Орешков В.И. Бизнес-аналитика: от данных к знаниям(+CD) / Учебное пособие. 2-е изд. СПб: Питер, 2013. 704 с.
12. Dahlskog S., Togelius J., Nelson M.J. Linear levels through n-grams // The 18th International Academic MindTrek Conference: Media Business, Management, Content & Services. 2014. P. 200-206.
13. Sutton R.S., Barto A.G. Reinforcement learning: An introduction. MIT Press, 1998. 291 p.
14. Беляев С.А., Михнович А.Г. Современные подходы к решению задачи стабилизации перевернутого маятника // Программные продукты, системы и алгоритмы. 2017. № 2. С. 2.