СОДЕРЖАНИЕ

Введение

Особый интерес к шагающим машинам появился в 1950-е годы после окончания второй мировой войны. Первой моделью «стопоходящей» машины была создана П.Л. Чебышевым. Способ передвижения с помощью ног, является наиболее распространенным в живой природе. Однако в технике он еще не получил заметного применения, прежде всего из-за сложности управления. Основными преимуществами шагающих роботов при сравнении с колесными роботами являются большая проходимость на пересеченной местности вплоть до возможности передвигаться прыжками и лазание по наклонным поверхностям [1].

Применение шагающих роботов актуально при работе в опасных зонах, в том числе при устранении последствий аварий техногенного характера и стихийных бедствий. В данных ситуациях необходимо исключить риск причинения вреда спасателям, а проходимости колесных роботов будет недостаточно. Главными проблемами алгоритмов управления шагающими роботами являются:

– сложность разработки алгоритма из-за большего числа решаемых подзадач;

– неустойчивость разработанного алгоритма к новым внешним условиям;

В данной работе будут рассмотрены основные подходы к созданию алгоритмов управления шагающими роботами, а также их сравнение. Цель сравнения – выбор подхода, благодаря которому будут решены указанные проблемы. Для выбранного подхода будет проведен анализ и сравнение существующих решений и выбор наилучших.

1 Анализ подходов к созданию автономных робототехнических систем

Для синтеза алгоритмов управления автономными роботами наиболее актуальными являются следующие подходы:

– эволюционные алгоритмы;

– подход на основе нечеткой логики;

– формальный подход;

– обучение с подкреплением.

Далее упомянутые подходы будут рассмотрены подробнее с целью выявить наилучший для решения задачи адаптивного управления шагающим роботом. Выбор будет происходить по следующим критериям:

– сложность формализации;

– возможность работать в неизвестной среде;

– сложность имплементирования;

– требовательность к вычислительным ресурсам.

Простота формализации является главным критерием, так как главная задача – упрощение существующих методов создания алгоритмов управления. Работа в заранее неизвестной среде обеспечивается адаптивностью алгоритма. Критерий сложности имплементирования определяет степень сложности воспроизведения алгоритма для новых начальных условий и конфигураций робота. Критерий требовательности к вычислительным ресурсам напрямую влияет на стоимость применения выбранного подхода.

1.1 Эволюционные алгоритмы

Эволюционные алгоритмы (ЭА) – это алгоритмы, созданные в качестве методов решения оптимизационных задач и основанные на принципах естественного отбора. ЭА моделируют базовые положения в теории биологической эволюции – процессы отбора (селекции), мутации и воспроизводства. Множество агентов, называемое популяцией, эволюционирует согласно правилам отбора в соответствии с целевой функцией. Таким образом, каждому агенту (индивидууму) популяции назначается значение его приспособленности (значение фитнесс-функции, являющейся

частным случаем целевой функции) в окружающей среде. Размножение и мутация позволяют изменяться агентам и приспособляться к среде [2].

В робототехнике применение ЭА позволяет выполнять различные практические задачи, в частности задачи обследования местности и маневрирования между препятствиями. Для агентов (роботов) описывается алгоритм, позволяющий им обмениваться генетической информацией с целью адаптации к решению конкретной задачи. В качестве генетической информации выступают различные стратегии управления приводами и системами робота на основе конкретных входных значений с датчиков. Такие стратегии могут задаваться изначально как случайно, так и согласно какой-либо эвристике. Цель – научить робота взаимодействовать с неизвестной средой.

Преимущества эволюционного подхода:

– возможность использования в задачах, сложно или полностью не поддающихся анализу и формализации;

– устойчивость в задачах с изменяющейся и/или частично неизвестной средой;

– существование большого количества как естественных, так и созданных человеком вариаций генетических операций.

Недостатки эволюционного подхода:

– оценка функции приспособленности (фитнесс-функции) для сложных проблем (задачи высокой размерности) требует больших вычислительных мощностей, что часто является фактором, ограничивающим использование алгоритмов искусственной эволюции;

– кодирование генетической информации (иногда называется геномом или хромосомами) представляет собой также сложный процесс, пусть и менее сложный, чем формализация всей задачи;

– симуляция на настоящих роботах требует большого объёма ресурсов, поэтому обычно синтез алгоритма производят в компьютерных симуляциях, а затем полученные алгоритмы встраиваются в настоящие роботы. Однако процесс создания самой симуляции также является трудоёмким.

1.2 Алгоритмы на основе нечёткой логики

Нечёткая логика – это раздел математики, являющийся обобщением логики и теории множеств, главным объектом исследования является нечеткое множество. В 1965 году Лотфи Заде расширил классическое понятие множества, допустив, что функция принадлежности принимает не только значения 0 и 1, а интервал от 0 до 1 [3]. Традиционный логический блок компьютера может давать определенный ответ – правда или ложь, что эквивалентно человеческим да или нет. Подход нечеткой логики позволяет давать более гибкие ответы, определяя аналоги человеческим: «возможно да», «не могу сказать», «возможно нет» и так далее.

Архитектура системы нечеткой логики состоит из четырёх основных частей:

– фузификатор (от английского fuzzy – нечеткий);
– база знаний;
– блок обработки;
– дефузификатор.

Фузификатор – модуль, трансформирующий входной сигнал в элемент нечеткого множества. Входом может быть сигнал измеренный сенсорами, например, температура или скорость. Обычно трансформированный сигнал делится на пять диапазонов:

– LP – вход большой положительный;
– MP – вход средний положительный;
– S – вход малый;
– MN – вход средний отрицательный;
– LN – вход большой отрицательный.

Функция, которая определяет отображение из пространства входов в интервал от 0 до 1, называется функцией принадлежности. В основном они используют 3 типа сигналов:

– импульсный;
– гауссовый;
– треугольный или трапецеидальный.

Пример треугольного сигнала для перевода напряжения в интервал от 0 до 1 приведен на рисунке 1.1.

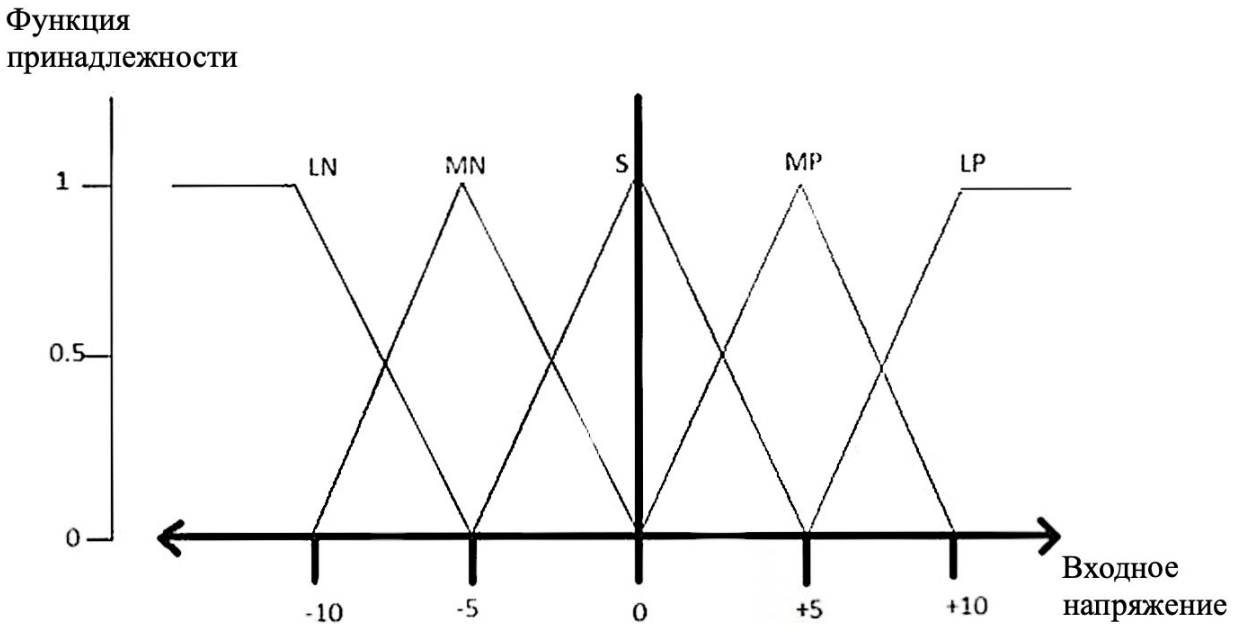


Рисунок 1.1 – Пример треугольного сигнала.

База знаний – база, созданная экспертами в той области, в которой реализуется система. В ней содержится набор условных правил.

Блок обработки – модуль, обрабатывающий входное значения, используя правила из базы знаний.

Дефузификатор трансформирует обработанный вход из нечеткой логики в классическую.

На рисунке 1.2 показана схема работы данной системы.

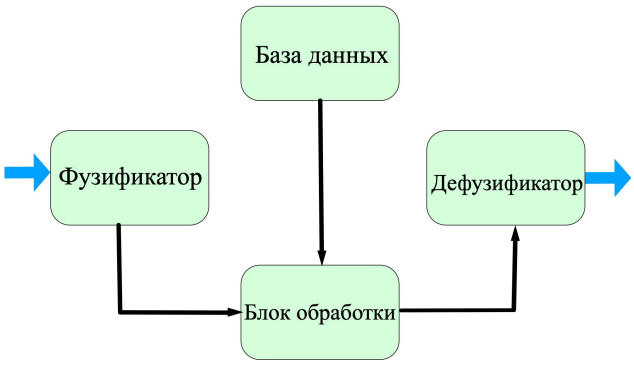


Рисунок 1.2 – Схема работы системы, использующей нечеткую логику.

Нечеткую логику используют в робототехнике для создания систем управления, систем стабилизации, экспертных систем и в случаях, когда необходимо сделать систему, реагирующую как человек, или для работы совместно с человеком [4], [5], [6].

Достоинства подхода, использующего нечеткую логику:

– подход устойчив к шуму входного сигнала;

– архитектура подхода понятна и проста в использовании;

– гибкость внесения изменений.

Недостатки подхода, использующего нечеткую логику:

– нет систематического подхода для построения архитектуры под каждую задачу;

– подход остается понятным, только когда архитектура не сложна;

– подход не дает высокой точности.

1.3 Формальный подход

Формализация – представление в виде формальной системы какой-либо содержательной информации, представленной на естественном языке, путём изъятия логических свойств элементов естественного языка, существенных отношений между этими элементами, а также определение принципов логической дедукции и критериев различия правильных способов рассуждения от неправильных.

Задача синтеза алгоритма управления шагающим роботом также требует формализации задачи. В этой задаче формализация заключается в составлении перечня всевозможных ситуаций, в которых может оказаться робот, и описание действий (и их порядка), которые необходимо в таких ситуациях предпринимать.

Преимущества:

– полная интерпретируемость алгоритма, т.е. возможность объяснить действия робота в любой ситуации;

– возможность оптимизации алгоритма управления.

Недостатки:

– требование значительных человеческих и вычислительных ресурсов для анализа и формализации задачи, а также для составления и оптимизации алгоритма (зачастую уникального, т.е. для которого сложно найти примеры имплементации).

1.4 Обучение с подкреплением

Обучение с подкреплением – это класс методов машинного обучения агента (робота), который не имеет сведений о среде, но имеет возможность производить какие-либо действия в ней. Действия агента переводят среду в новое состояние и агент получает от среды некоторое вознаграждение или наказание в соответствии с функцией подкрепления. Функция подкрепления определяет цель в процессе обучения с подкреплением и является по сути соответствием между состояниями среды и числом, подкреплением, показывающим желательность, ценность состояния. В качестве функции подкрепления может быть, например, расстояние от робота до определённой точки пространства; высота верхней точки шагающего робота (косвенно сообщающая о том, что робот не упал); уровень заряда батареи бортового аккумулятора. В данном классе методов большое внимание уделяется поощрению или наказанию не только текущих действий, которые непосредственно привели к положительному или отрицательному подкреплению, но и предшествующих им. Поэтому в качестве целевой функции выступает сумма подкреплений на определённом промежутке пространства, например, суммарное подкрепление за прохождение определённой траектории. В то время как функция подкрепления определяет прямую, характерную желательность состояния среды, целевая функция обнаруживает долгосрочную желательность состояний после принятия во внимание состояний, которые последуют за текущим, и подкреплений, соответствующих этим состояниям. Например, состояние может повлечь низкое непосредственное подкрепление, но при этом сильно положительно повлиять на суммарную оценку, потому как за ним регулярно следуют другие состояние, которые приносят высокие подкрепления. Единственная цель агента состоит в максимизации итогового

подкрепления (целевой функции), которое тот получает в процессе длительной работы [7].

В робототехнике обучение с подкреплением активно применяется в задаче передвижения мобильного робота по лабиринту [8] и в задаче управления манипулятором [9]. Также существуют попытки обучать агента выполнять задачи разного плана при помощи одного алгоритма[10].

Преимущества обучения с подкреплением:

– обучившаяся нейросеть позволяет решать общую задачу;

– для обучения нейросети требуется меньше вычислительных мощностей, чем в эволюционных методах;

– достаточно одного агента для обучения.

Недостатки обучения с подкреплением:

– требования к вычислительным ресурсам в случаях многоразмерных задач;

– в процессе обучения не всегда находится оптимальная стратегия для получения максимальной долгосрочной награды.

1.5 Вывод по главе

В данной главе были рассмотрены наиболее актуальные подходы для решения задачи синтеза алгоритма управления – подход на основе нечеткой логики, эволюционный подход, подход на основе обучения с подкреплением, экспертный подход. Рассмотренные алгоритмы были сравнены между собой по критериям – степень формализации, обобщающая способность, трудозатраты и требовательность к вычислительным ресурсам. Результаты сравнения представлены в таблице 1.

Таблица 1 – Сравнение подходов к созданию систем управления

Критерий	Нечеткая логика	Формальный подход	Эволюционные алгоритмы	Обучение с подкреплением
Сложность формализации	Высокая	Высокая	Средняя	Малая
Обобщающая способность	Нет	Нет	Да	Да
Трудозатраты	Средне	Много	Средне	Мало
Требовательность к вычислительным ресурсам	Мало	Мало	Много	Средне

Степень формализации – основной критерий, он показывает количество начальной информации, которую необходимо знать об агенте, и как полно необходимо описать кинематику и динамику агента. В эволюционном подходе и обучении с подкреплением происходит абстрагирование от физических характеристик агентов, благодаря этому трудозатраты на создание алгоритмов управления для сложных систем сокращаются. Также из-за того, что в этих методах не используется описание конкретной задачи, а только функция содержащая общие аспекты работы алгоритма, то и сам алгоритм имеет большую обобщающую способность.

Для работы эволюционных методов необходима симуляция множества агентов, что сильно увеличивает требования к вычислительным ресурсам. Описание целевой функции в эволюционных методах и в обучении с подкреплением – относительно простая задача. Однако кодирование генома агента в эволюционных алгоритмах – несравненно больший труд.

В связи с вышесказанным, лучшим из рассмотренных методов по выбранным критериям является метод, использующий обучение с подкреплением.

2 Анализ методов обучения с подкреплением

Для решения задачи синтеза алгоритма управления был выбран подход с использованием обучения с подкреплением. Алгоритмы обучения с подкреплением делятся на две группы:

– с аналоговым выходом;

– с дискретным выходом.

Данные группы алгоритмов решают разные задачи: первые используются в системах, где управляющий сигнал представлен диапазоном значений, например, управление напряжением двигателя. Вторая группа алгоритмов используется в системах, где управляющий сигнал представлен конечным набором действий.

В этой главе будут рассмотрены 3 алгоритма с аналоговым выходом:

– DDPG;

– PPO;

– TRPO.

А также 4 алгоритма с дискретным выходом:

– CEM;

– DQN;

– DUEL DQN;

– SARSA.

Вышеупомянутые алгоритмы сравнены между собой по критериям следующим критериям:

– скорость обучения;

– качество;

– стабильность;

– количество обучаемый параметров.

2.1 CEM

Метод перекрестной энтропии (CEM) – общий алгоритм для решения глобальной оптимизационной задачи. Данный метод позволяет обучить нейронную сеть для решения задачи обучения с подкреплением. Нейронная сеть является управляющим звеном, на нее поступают выходы агента,

например, показания датчиков, а выходом нейронной сети являются входы агента, то есть стратегия управления.

На практике стратегия представляется, как распределения вероятности над действиями. Данный подход очень похож с задачей классификации: выбор одного действия из набора. Алгоритм пропускает через нейронную сеть наблюдение окружающей среды, забирает вероятностное распределение над действиями (стратегию) и генерирует случайную выборку из этого распределения. Это случайная выборка добавляет неопределенности агенту, что хорошо влияет на начало обучающего процесса. Чем дольше происходит процесс обучения, тем меньше становится неопределенность. После действия, агент получает награду и цикл повторяется [11].

В процессе обучения мы минимизируем разницу между начальной стратегией и оптимальной, найденной методом выборки по значимости [12], в итеративном процессе. Разность между текущей и оптимальной стратегией определяется дивергенцией Кульбака-Лейблера – формула (2.1).

$$D_{\mathrm{KL}}(P\|Q) = \int_X p \, \log \frac{p}{q} \, \mathrm{d}\mu \tag{2.1}$$

Где p(x), q(x) – функции плотности распределения, сравниваемых величин.

Достоинства данного метода:

– хорошо работает в легких средах;

– робастный к изменению гиперпарамтеров.

Недостатки CEM:

– низкая скорость обучения.

2.2 DQN

Deep Q – Network (DQN) – улучшение классического алгоритма обучения с подкреплением Q-learning за счёт использования нейронных сетей.

Q-learning (в русскоязычной литературе иногда употребляется Q-обучение) – алгоритм, согласно которому агент на основе получаемого от среды

вознаграждения формирует функцию полезности Q, что впоследствии дает ему возможность уже не случайно выбирать стратегию поведения, а учитывать опыт предыдущего взаимодействия со средой. Одно из преимуществ Q-обучения – то, что оно в состоянии сравнить ожидаемую полезность доступных действий, не формируя модели окружающей среды. Применяется для ситуаций, которые можно представить в виде Марковского процесса принятия решений (англ. Markov decision process (MDP)), то есть процесса, в котором вероятности переходов между состояниями не зависят от истории предыдущих переходов.

Алгоритм Q-learning пытается получить значение функции Q в явном виде для всех возможных пар «состояние среды - действие», чтобы потом максимизировать ожидаемый результат, найдя оптимальную стратегию. Однако даже в самых простых задачах робототехники число состояний среды очень велико, а если его умножить на количество возможных действий, то получится таблица, обрабатывать которую на борту робота будет затруднительно. Поэтому в алгоритме DQN таблица Q заменена функцией, которую приближает свёрточная нейронная сеть. Помимо этого, у DQN есть ещё несколько усовершенствований по сравнению с классическим алгоритмом Q-learning. Во-первых, практика показывает, что обучаться непосредственно на состояниях, происходящих друг за другом, – плохой подход, так как такие состояния слишком похожи друг на друга, сильно коррелируют, причём со временем их распределение, естественно, сдвигается в зависимости от действий (или, например, положения) робота, но остается локализованным. Это мешает эффективному обучению, ведь в обычной постановке задачи обучения предполагается, что тренировочные данные независимы, а распределение данных со временем не меняется. Поэтому по мере обучения DQN сначала накапливает некоторый опыт, сохраняя свои действия и их результаты на протяжении какого-то времени, а потом выбирает из этого опыта случайную выборку (mini-batch) отдельных примеров для обучения, взятых в случайном порядке. Во-вторых, важную роль для успеха сыграло то, что при обучении DQN сеть, которая отвечала за целевую функцию, была отделена от обучающейся сети. Сделано это по причине того, что на практике

нейронные сети довольно быстро заходят в локальные экстремумы и начинают очень глубоко исследовать глобально неважные части пространства поиска, бесцельно тратя ресурсы и фактически не обучаясь. Избежать этого можно, сделав так, чтобы сеть не сразу использовала обновленную версию в целевой функции, а обучалась достаточно долгое время по старым образцам, прежде чем использовать обновлённую целевую функцию. В-третьих, на практике обычно применяется архитектура, в которой возможное действие агента не подается на вход, а просто у сети столько выходов, сколько возможных действий, и, получая на входе состояние, сеть пытается предсказать результаты полезности, ценности каждого действия (после чего, естественно, выбирает максимальный). Это важное улучшение, потому что оно позволяет получить ответы сразу для всех действий за один проход по сети, что ускоряет происходящее в разы, а сеть от этого сильно сложнее не становится, ведь основная часть ее «логики» остается прежней и используется заново.

Преимущества:

– простой относительно других алгоритмов глубокого обучения с подкреплением, в следствие чего обучается быстрее более сложных алгоритмов;

– благодаря усовершенствованиям справляется с проблемой локальных минимумов.

Недостатки:

– не использует всю историю обучения;

– может ограничивать скорость обучения.

2.3 DUEL DQN

Duel DQN является улучшением алгоритма DQN за счёт разделения Q-сети на два канала, один из которых, вычисляет оценку позиции V, которая является функцией только состояния среды, а другой – зависящую от действия функцию преимущества (advantage function) A. На последнем этапе они просто складываются в прежнюю функцию Q. Таким образом, одна часть сети обучается оценивать позицию как таковую, а другая – предсказывать, насколько полезны будут разные наши действия в этой позиции. Такое небольшое

изменение в архитектуре сети приближает смысл оценки к естественному, что часто существенно улучшает результаты [13]. Сравнение архитектур DQN и DUEL DQN показана на рисунке 2.1.

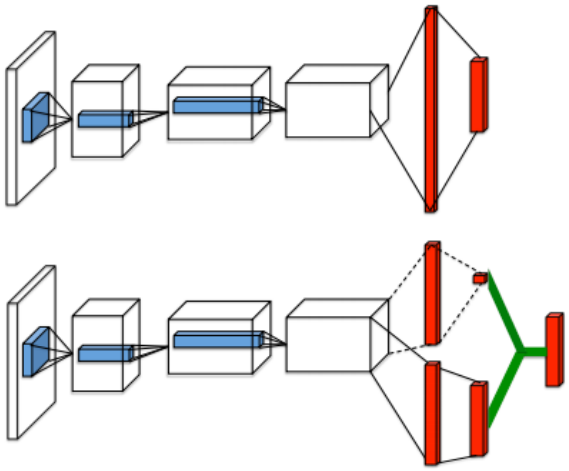


Рисунок 2.1 – Сравнение архитектур DQN(сверху) и DUEL DQN(снизу).

2.4 SARSA

Алгоритм SARSA – схож с алгоритмом Q-learning. Данный алгоритм также выбирает действие у которого будет максимальное значение Q. Главным отличием является то, что SARSA вычисляет значение Q используя текущую стратегию, а Q-learning жадный поиск. Данное различие позволяет использовать этот алгоритм для обучения нейросети на реальном дорогостоящем роботе, так как каждое выбранное действие не будет кардинально отличаться от предыдущего.

Достоинства:

– учитывает штрафы от предыдущих действий.

Недостатки:

– может игнорировать оптимальную стратегию;

– учиться дольше чем Q-learning.

2.5 DDPG

В алгоритмах Q-learning, DQN, Duel DQN и SARSA действия выбираются согласно максимальному Q-значению. В таких алгоритмах (они называются value-based, то есть основанных на значениях функции Q) стратегия

представляет собой не что иное, как выбор действия с максимальным значением Q. Проблема value-based подхода заключается в необходимости вычисления оценки награды для каждого возможного действия в каждом возможном состоянии. Особенно остро эта проблема встаёт, когда дело касается бесконечного пространства действий (обычно это физические задачи, например, классическая задача балансировки обратного маятника. Также это задачи управления вращением колёс, движением манипулятора). Например, DQN создавался для решения проблемы с достаточно высокоразмерным пространством состояний (пиксели на экране в играх Atari), но при этом очень малоразмерным дискретным пространством действий (до двух десятков комбинаций кнопок). Но задачи физического контроля имеют непрерывные и высокоразмерные пространства действий. DQN не может быть непосредственно применён к непрерывным пространствам даже при грубой дискретизации пространства действий, так как размер выходного слоя нейронной сети Q-network растёт экспоненциально с ростом степеней свободы агента [14].

В алгоритмах градиентного спуска по стратегиям (policy gradient (PG) methods) вместо обучения функции Q, возвращающей ожидаемую ценность каждого действия в каждом состоянии, напрямую обучается стратегия, которая состоянию ставит в соответствие действие. В данном семействе алгоритмов центральную роль занимает стратегия, согласно которой агент совершает действие в зависимости от текущего состояния и от параметров самой стратегии (обычно это веса нейронной сети). Задача обучения с подкреплением заключается в максимизации целевой функции. Вознаграждение, которое напрямую влияет на значение целевой функции, зависит от стратегии, которая в свою очередь зависит от своих параметров, следовательно, сама целевая функция зависит от параметров стратегии. PG, который является на данный момент центральным методом обучения с подкреплением в задачах робототехники, базируется на идее оптимизации целевой функции методом градиентного спуска по стратегиям вместо построения моделей окружающей среды или оценки функции Q [15].

Такой подход работает не в каждой задаче, так как, используя разумные ресурсы, можно посчитать градиент только за ограниченное количество шагов и с ограниченной точностью. Однако благодаря policy gradient theorem производную можно вычислять для более продолжительных промежутков, например, для целой траектории. На наборах траекторий и суммарной награды по итогу прохождения соответствующих траекторий обучаются параметры стратегии.

Стратегия бывает двух видов: детерминированная и стохастическая. Детерминированная состоянию ставит в соответствие действие, которое необходимо в этом состоянии выполнить. Детерминированные стратегии используются в детерминированных средах или в средах, где случайные события слабо влияют на поведение агента (например, движение исправного робота в лабиринте при стабильных погодных условиях и при однородном освещении). Стохастические стратегии состоянию среды ставят в соответствие распределение над действиями. Это означает, что разные действия могут быть выбраны для одного и того же состояния. Такие стратегии используются в средах, где доля случайных или неопределённых событий значительна, то есть почти в любой в естественной среде. Детерминированные стратегии проще описываются и обучаются, поэтому алгоритмы deterministic policy gradient (DPG) более популярны, нежели stochastic policy gradient (SPG).

На основе идей, принёсших успех DQN, был создан алгоритм deep deterministic policy gradient (DDPG) [16], применимый для непрерывного пространства действий. Также важной чертой DDPG является применение архитектуры «actor-critic» (устоявшегося русского названия не существует, но приблизительный перевод – «действующее лицо-критик»), которая является по сути объединением идей Q-learning и policy gradient, показано на рисунке 2.2.

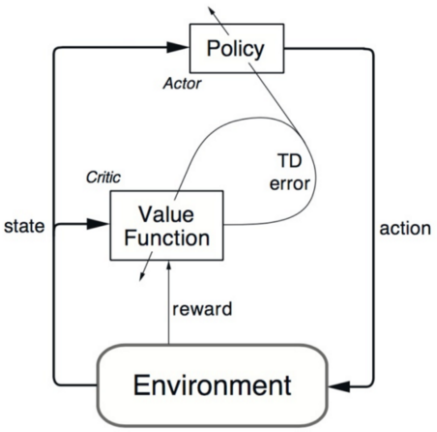


Рисунок 2.2 – Архитектура «actor-critic».

В данном алгоритме обучаются две нейронные сети: сеть-actor обучает параметры стратегии, чтобы поданному на вход состоянию ставилось в соответствие лучшее действие, и сеть-critic, обучающаяся оцениванию стратегии по ошибке, которая высчитывается по тому же правилу, что и при обновлении Q-функции в Q-learning. Но при этом функция оценки используется только для оптимизации параметров стратегии, но сама при этом при выборе действия не используется.

Преимущества алгоритма DDPG:

– Алгоритмам, основанным на оптимизации функции оценки, присущи большие колебания во время обучения. Эти колебания связаны с тем, что выбор действия может резко измениться при сколь угодно малых изменениях оценки действий. В алгоритмах, основанных на оптимизации стратегии, ищется градиент по стратегии, поэтому процесс обучения проходит более гладко.

– Возможность рассмотрения проблемы исследования (problem of exploration) независимо от алгоритма DDPG.

– Возможность работать с высокоразмерными непрерывными пространствами действий.

Недостатки алгоритма DDPG:

– Вычисление градиента по стратегии приводит к возможной сходимости в локальном максимуме вместо глобального. Такая проблема

отсутствует в value-based подходе, где постоянно ищется лучшее из возможных действий.

– В случае отсутствия добавленной извне формализации исследования среды, алгоритм не будет исследовать среду вообще.

2.6 TRPO

Метод региона доверия (TRPO) – градиентный метод оптимизации, позволяющий напрямую изменять стратегию. Для нахождения оптимальной стратегии, необходимо максимизировать функцию наград. На качество оценки ожидаемого значения функции наград с текущей стратегией влияет разность между текущей стратегией и предыдущей. Если стратегия сильно отличается от предыдущей, то точность будет ниже. Для увеличения точности вводится понятие региона доверия, в котором гарантировано качество стратегии будет не ухудшаться. Оптимальная стратегия ищется итеративным процессом, каждый раз выбирается из региона доверия [17].

Для гарантии не ухудшения стратегии используется итеративный подход минимальная-максимизация [18]. Регионом доверия является регион с определенным радиусом, в котором ищется оптимальная точка. Радиус может меняться от шага к шагу. Для нахождения градиента с использованием предыдущих данных используется выборка по значимости.

Достоинства данного метода:

– легкая масштабируемость,

– стабильность

Недостатки данного метода:

– Тяжело использовать в случаях большого количества выходов.

2.7 PPO

Proximal Policy Optimisation (PPO) – метод обучения для задачи обучения с подкреплением, на каждом шаге максимизируя функцию наград, при этом не сильно изменяя стратегию. Данный метод схож с методом TRPO, но есть и отличия. Во-первых, для вычисления второй производной, метод TRPO вычисляет приближенное значение второй производной, а метод PPO заменяет на первую производную, за счёт введения гиперпараметров. Также

для нахождения новой стратегии в регионе доверия, используется дивергенция Кульбака-Лейблера между текущей стратегией и новой [19]. Достоинства данного метода:

– хорошо работает с большим количеством выходов.

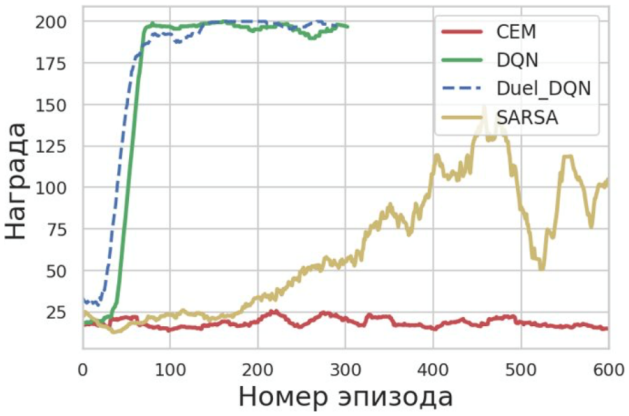
Недостатки:

– менее устойчивый чем TRPO.

2.8 Эксперименты

Для сравнения алгоритмов обучения с подкреплением были проведены эксперименты, отдельно для дискретного управления и отдельно для аналогового. Для дискретного управления решается задача равновесия обратного маятника, а для аналогового управления решается задача равновесия двойного обратного маятника. Выбранные задачи обусловлены наличием готовых примеров. Симуляция алгоритмов происходит в среде Mujoco [20]. Эта среда выбрана, благодаря быстрой обработке кинематики и динамики объектов. Оба алгоритма будут сравнены по одинаковым критериям. Скорость обучения – определяется по эпизоду, на котором алгоритм достигает максимальной награды. Чем меньше это значение, тем быстрее алгоритм обучается. Лучший результат определяет качество работы алгоритма. Чем выше значение данного критерия, тем лучше качество. Стабильность оценивается по графику зависимости шага от награды, зависимость стабильная, если не происходит переобучения с увеличением продолжительности обучения. На рисунке 2.3, слева и справа показаны полученные зависимости награды от шага, для аналогового и дискретного управления соотвественно. Количество обучаемых параметров – это количество свободных параметров нейронной сети.

Для дискретного управления



Для аналогового управления

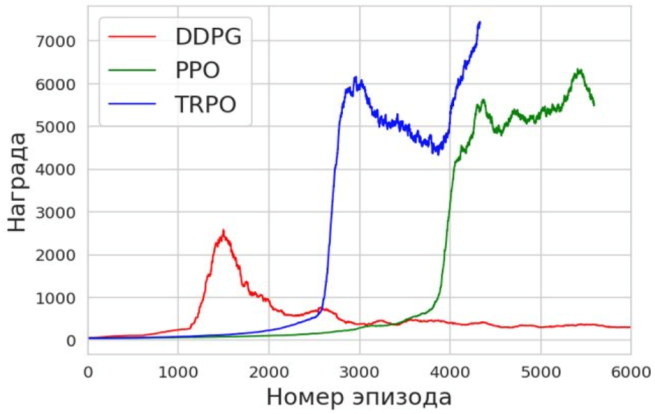


Рисунок 2.3 – Зависимость награды от номера эпизода для аналогового и дискретного управления.

2.9 Вывод по главе

По результатам экспериментов составлена сравнительная таблица 2.

Таблица 2 – Сравнение алгоритмов обучения с подкреплением

Алгоритм	Скорость обучения	Лучший результат	Стабильность	Количество обучаемых параметров	Тип управления
CEM	2500	35	Да	658	Дискретное
DQN	80	200	Да	658	Дискретное
DUEL DQN	140	200	Да	658	Дискретное
SARSA	450	160	Нет	658	Дискретное
DDPG	1500	2800	Нет	9000	Аналоговое
PPO	4200	7500	Да	4485	Аналоговое
TRPO	3000	6200	Да	4485	Аналоговое

Из полученных результатов, лучшими методами обучения для дискретного управления является DQN и Duel DQN. Алгоритмы достигли лучшего результата в 200 шагов, что является максимальным в данном эксперименте, также эти алгоритмы стабильные в отличие от алгоритма SARSA. Особенностью алгоритма SARSA является небольшое отклонение

новой стратегии от предыдущей, а так как количество состояний движения было всего 2, то заданный алгоритм пытался уравновесить маятник, используя только движение в одну из сторон. Алгоритм DQN обучился быстрее, ему потребовалось на 80 эпизодов меньше, чем Duel DQN.

Для аналогового управления самую большую скорость обучения показал алгоритм DDPG, но его максимальный результат в 2,2 раза меньше, чем результат TRPO, и в 2,7 раза меньше, чем результат PPO. Алгоритмы PPO и TRPO не были переобучены за 4500 эпизодов, в то время как DDPG начал переобучаться сразу после своего максимального значения, после 1500 эпизодов. Лучшими алгоритмами для аналогового управления являются алгоритмы PPO и TRPO, если основным является критерий качества, а если главным является скорость обучения, то DDPG.

Заключение

В данной работе были рассмотрены подходы к созданию алгоритмов управления шагающими роботами, с целью сравнения и выбора лучшего для решения следующих проблем:

– уменьшение сложности разработки алгоритма;

– увеличение устойчивости разработанного алгоритма к новым внешним условиям;

Были рассмотрены наиболее актуальные подходы для решения задачи управления: подход на основе нечеткой логики, эволюционный подход, подход на основе обучения с подкреплением, экспертный подход. Результаты сравнения представлены в таблице 1. Степень формализации – основной критерий, он показывает количество начальной информации, которую необходимо знать об агенте, и как полно необходимо описать кинематику и динамику агента. Лучшим подходом в данном сравнение является алгоритм на основе обучения с подкреплением.

Для выбранного подхода были рассмотрены алгоритмы обучения для дискретного и аналогово выхода. С аналоговым выходом:

– DDPG;

– PPO;

– TRPO.

С дискретным выходом:

– CEM;

– DQN;

– DUEL DQN;

– SARSA.

Для сравнения алгоритмов обучения с подкреплением были проведены эксперименты, отдельно для дискретного управления и отдельно для аналогово. Для дискретного управления решается задача равновесия обратного маятника, а для аналогово управления решается задача равновесия двойного обратного маятника. Выбранные задачи обусловлены наличием готовых примеров.

Симуляция алгоритмов происходит в среде Mujoco [20]. Данная среда выбрана, благодаря быстрой обработке кинематики и динамики объектов.

Данные алгоритмы были сравнены между собой по критериям скорость обучения, качество, стабильность, количество обучаемый параметров. Результаты сравнения приведены в таблица 2. Из полученных результатов лучшими методами обучения для дискретного управления являются DQN и Duel DQN. Лучшими алгоритмами для аналогового управления являются алгоритмы PPO и TRPO, если основным является критерий качества, а если главным является скорость обучения, то DDPG.

Для дальнейшей работы необходимо создать модель шагающего робота и протестировать выбранные алгоритмы обучения на ней. Целью будет являться нахождения лучшего алгоритма для задач перемещения.

Список использованных источников

1 Е.И.Юревич «Основы робототехники», БХВ-Петербург, – 304 с. 2017.

2 L.Silva, A.Silva «An Evolutionary Algorithm for Autonomous Robot Navigation», The International Conference on Computational Science, 2016.

3 Л.Заде «Понятие лингвистической переменной и его применение к принятию приближенных решений», М.: Мир, – 166 с. 1976.

4 М.С.Ситников «Анализ и синтез интеллектуальных систем автоматического управления с нечеткими регуляторами», Москва, 2008.

5 A.Eid, R.Abdel-Fadil «Fuzzy logic control of modern aircraft actuators», 3rd International Conference on Energy Systems and Technologies, 2015.

6 P.Lamkhade, C.B.Kadu «Design and implementation of fuzzy logic controller for level control», 3rd International Conference on Energy Systems and Technologies, 2015.

7 Обучение с подкреплением, [Электронный ресурс]. – URL: http://apsheronsk.bozo.ru/Neural/Lec8.htm (Дата обращения: 12.01.2019).

8 I.Zamora, N.G.Lopez «Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo», arXiv:1608.05742v2, 2016.

9 E.Tzeng, C.Devin «Adapting Deep Visuomotor Representations with Weak Pairwise Constraints», arXiv:1511.07111v5, 2015.

10 C.Florensa, D.Held «Automatic Goal Generation for Reinforcement Learning Agents», arXiv:1705.06366, 2018.

11 D.P.Kroese «Cross-Entropy Method», arXiv:1503.01842v1, 2015.

12 S.T.Tokdar, R.E.Kass «Importance Sampling: A Review», Department of Statistics, Carnegie Mellon University, Pittsburgh, 2008.

13 V. Mnih «Human-Level Control through Deep Reinforcement Learning», journal Nature, 2015.

14 С.И.Николенко «Глубокое обучение. Погружение в мир нейронных сетей», Питер, – 480 с. 2018.

15 Policy gradient methods, [Электронный ресурс]. – URL: https://medium.freecodecamp.org/an-introduction-to-policy-gradients-with-cartpole-and-doom-495b5ef2207f (Дата обращения: 15.01.2019).

16 T.P.Lillicrap, J.J.Hunt «Continuous control with deep reinforcement learning», arXiv:1509.02971, 2015.

17 J.Schulman, S.Levine «Trust Region Policy Optimization», arXiv:1502.05477v5, 2017.

18 P.Bansal, E.Guerra «Minorization-Maximization (MM) algorithms for semiparametric logit models: Bottlenecks, extensions, and comparisons», School of Civil and Environmental Engineering, Cornell University, USA, 2018.

19 J.Schulman, F.Wolski, P.Dhariwal, A.Radford, O.Klimov «Proximal Policy Optimization Algorithms», arXiv:1707.06347v2, 2017.

20 MuJoCo – documentation, [Электронный ресурс]. – URL: http://www.mujoco.org/book/index.html (Дата обращения: 12.01.2019).