МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ» (ФГБОУ ВО «ПГТУ»)

Кафедра иностранных языков и лингвистики

Микриуправление в Starcraft с использованием обучения с подкреплением и постепенного трансферного обучения

Перевод

Kun Shao, Yuanheng Zhu, Member, IEEE and Dongbin Zhao, Senior Member, «StarCraft Micromanagement with Reinforcement Learning and Curriculum Transfer Learning», принята IEEE Transactions on Emerging Topics in Computational Intelligence, 3 апреля 2018 г, [Электронный ресурс] URL: https://arxiv.org/pdf/1804.00810.pdf

Выполнил: студент гр.ПСм-21 Колчин И. А. Проверил: к.пед.н., доцент кафедры ИЯиЛ О.В. Филипчук.

Йошкар-Ола

2020

Оглавление

[Аннотация 3](#_Toc30692386)

[Оригинал 3](#_Toc30692387)

[Перевод 3](#_Toc30692388)

[Перевод статьи 5](#_Toc30692389)

[1. Введение 5](#_Toc30692390)

[2. Постановка проблемы и предпосылки 5](#_Toc30692391)

[2-a. Постановка проблемы 6](#_Toc30692392)

[2-b. Обучение с подкреплением 6](#_Toc30692393)

[2-c. Постепенное трансфертное обучение 6](#_Toc30692394)

[3. Обучение модели микроменеджменту 6](#_Toc30692395)

[3-a. Представление многомерного состояния 6](#_Toc30692396)

[3-b. Определение действий 6](#_Toc30692397)

[3-c. Архитектура нейросети 6](#_Toc30692398)

[4. Метод обучения микроменеджменту 6](#_Toc30692399)

[4-a. Много-агентный алгоритм с градиентным спуском и разделением параметров Сарса(λ) 6](#_Toc30692400)

[4-b. Функция вознаграждения 6](#_Toc30692401)

[4-c. Пропуск кадров 6](#_Toc30692402)

[5. Постановка эксперимента 6](#_Toc30692403)

[5-a. Сценарии микроуправления в Starcraft 6](#_Toc30692404)

[5-b. Обучение нейросети 6](#_Toc30692405)

[6. Результаты и обсуждения 6](#_Toc30692406)

[6-a. Микроуправление малого масштаба 6](#_Toc30692407)

[6-a1. Голиафы против Зилотов 6](#_Toc30692408)

[6-a2. Голиафы против Зерглингов 6](#_Toc30692409)

[6-b. Микроуправление большого масштаба 6](#_Toc30692410)

[6-b1. Морпехи против Зерглингов 6](#_Toc30692411)

[6-c. Анализ стратегий 6](#_Toc30692412)

[6-c1. Разделение врагов 6](#_Toc30692413)

[6-c2. Сохранение команды 7](#_Toc30692414)

[6-c3. Бей и беги 7](#_Toc30692415)

[6-c4. Существующие проблемы 7](#_Toc30692416)

[7. Заключение и будущие работы 7](#_Toc30692417)

[Приложения 7](#_Toc30692418)

[Терминологический словарь 7](#_Toc30692419)

# Аннотация

Стратегические игры в реальном времени является важной областью игр с искусственным интеллектом в последние годы. В этой статье представлен метод обучения с подкреплением и метод постепенного трансферного обучения, с помощью которых осуществляется управление множеством боевых единиц в StarCraft. Мы определяем эффективное представление состояний, которое устраняет сложность, вызванную большим пространством состояний в игровой среде. Затем для обучения юнитов предлагается алгоритм с разделением параметром между агентами с градиентным спуском Сарса (λ лямбда) (PS-MAGDS). Политика обучения распространяется среди наших юнитов, чтобы поощрять совместное поведение. Мы используем нейронную сеть в качестве аппроксиматора функции для оценки функции «действие-значение» и предлагаем функцию вознаграждения, которая помогает юнитам сбалансировать их движение и атаку. Кроме того, метод трансферного обучения используется для расширения нашей модели до более сложных сценариев, что ускоряет процесс обучения и повышает эффективность обучения. В мелкомасштабных сценариях наши юниты успешно учатся сражаться и побеждать встроенный ИИ со 100% вероятностью выигрыша. В крупномасштабных сценариях метод постепенного обучения используется для постепенного обучения группы юнитов и показывает превосходную производительность по сравнению с некоторыми базовыми методами в целевых сценариях. Благодаря обучению с подкреплением и постепенному трансферному обучению наши юниты способны выучить подходящие стратегии в сценариях микроуправления StarCraft.

Ключевые слова: обучение с подкреплением, трансферное обучение, обучение по плану(или постепенное обучение), нейронная сеть, игровой ИИ.

# Введение

Искусственный интеллект значительно улучшился за последнее десятилетие. Являясь отличным испытательным полигоном для исследований ИИ, игры помогают ИИ развиваться с самого его зарождения, включая древние настольные игры (го и гомоку)[1], [2], [3], [4], классические видео игры Атари, и игры с неполной информацией(в 7 говорится об ИИ для покера). Эти игры имеют фиксированный, ограниченный набор действий и исследователи могут управлять только одним агентом в игровой среде. С другой стороны, существует большое количество игр, включающих несколько агентов и требующих сложных правил, которые намного сложнее для исследования ИИ.

В этой статье, мы фокусируемся на стратегиях реального времени для исследования обучения мультиагентному управлению. RTS игры обычно запускаются в режиме реального времени, что отличается их от настольных игр, где ходят по очереди[8]. Как в одной из самых популярных RTS игр, в Starcraft огромная база игроков и многожество профессиональных соревнований, требующих различных стратегий, тактик и методов реактивного управления. Для исследования игрового ИИ StarCraft предоставляет идеальную среду для изучения управления несколькими юнитами с различными уровнями сложности [9]. В последние годы исследования ИИ для StarCraft достигли впечатляющих успехов благодаря соревнованиям ИИ для StarCraft и интерфейсу для прикладного программирования для Starcraft Brood War (BWAPI) [10]. Недавно исследователи разработали более эффективные платформы для содействия развитию этой области, включая TorchCraft, ELF и PySC2. StarCraft ИИ направлен на решение ряда задач, таких как пространственные и временные закономерности(рассуждения), совместная работа нескольких агентов, моделирование оппонентов и планирование состязаний [8]. В настоящее время не разработано ни одного игрового ИИ для полноценной игры в StarCraft на основе методов машинного обучения. Многие исследователи фокусируются на микроуправлении как на первом этапе изучения ИИ в StarCraft [11]. В боевых сценариях юниты должны перемещаться в высоко динамичной среде и атаковать врагов в радиусе атаки. Существует много методов для микроуправления StarCraft, включая потенциальные поля для пространственной навигации и обхода препятствий [12], [13], байесовское моделирование для решения проблем неполной информации и неопределенности в игре [14], эвристический поиск по дереву игр для обработки последовательности заказов юнитов и зданий, и управления юнитами [15], а также нейроэволюция для управления отдельными юнитами с рукотворными улучшениями [16].

Как интеллектуальный метод обучения, обучение с подкреплением (RL) очень подходит для последовательных задач принятия решений. В микроуправлении StarCraft есть несколько интересных приложений с методами RL. Шантия и соавт. Используют алгоритм онлайн Сарса и нейронную версию Сарсы с функцией кратковременного поощрения, чтобы контролировать нападение и отступление юнитов [17]. Они используют сетку видимости для получения информации о местности. Этот метод требует ручной разработки, а количество входных узлов должно меняться в зависимости от количества единиц. С другой стороны, они применяют метод инкрементного обучения, чтобы масштабировать задачу до более крупного сценария с 6 юнитами. Тем не менее, процент выигрышей при инкриментном обучении все еще ниже 50%. Вендер и соавт. используют различные RL-алгоритмы в микроуправлении, в том числе Q-обучение и Сарса [18]. Они управляют одним мощным юнитом, чтобы играть против нескольких слабых юнитов, без сотрудничества и совместной работы со своими юнитами.

В последние несколько лет глубокое обучение достигло замечательных результатов во многих сложных задачах [19] и значительно улучшило обобщение и масштабируемость традиционных RL-алгоритмов[5]. Глубокое обучение с подкреплением (DRL) может научить агентов принимать решения в пространстве состояний высокой размерности сквозным/непосредственным методом. Усиниер и соавторы предлагают RL-метод для решения проблемы микроуправления с помощью глубокой нейронной сети [20]. Они используют жадный MDP для последовательного выбора действий для юнитов на каждом временном шаге с оптимизацией нулевого порядка для обновления модели. Этот метод способен контролировать всех юнитов, принадлежащих игроку, и наблюдать за полным состоянием игры.

Пэнг и соавт. используют метод критики агентов и рекуррентные нейронные сети (RNN), для военных игр в StarCraft [21]. Зависимость юнитов моделируется двунаправленной рекуррентной нейронной сетью в скрытом слое, и обновление её градиента эффективно распространяется по всей сети. Отличие работ Усиниер и Пэнг в том как реализуется централизация контроллеров, Фоэстер и соавторы предлагают многоагентный метод с критикой агентов для решения децентрализованных задач микроуправления, который значительно повышает производительность по сравнению с централизованными RL -контроллерами[22].

Для микроуправления StarCraft традиционные методы испытывают трудности при обработке сложного пространства состояний и действий, а также в обучении тактике сотрудничества. Современные методы основаны на сильных вычислительных возможностях, которые дает глубокое обучение. Кроме того, для изучения микроуправления с помощью безмодельных RL методов, как правило, требуется много времени на обучение, что является еще более серьезной проблемой в крупномасштабных сценариях. В этой статье мы рассматриваем более эффективное представление состояний, чтобы убрать сложность, вызванную большим пространством состояний, и предлагаем соответствующий RL-алгоритм для решения проблемы многоагентного принятия решений в микроуправлении StarCraft. К тому же мы вводим постепенное трансферное обучение, чтобы расширить RL-модель для различных сценариев и повысить эффективность выборки.

Основные новшества показаны в трех частях. Во-первых, мы предлагаем эффективный метод представления состояний для работы с большим пространством состояний в микроуправлении StarCraft. Этот метод учитывает атрибуты юнитов и расстояния между ними, что позволяет иметь произвольное количество юнитов обеим сторонам. По сравнению с аналогичными работами наше представление состояния значительно меньше и эффективнее. Во-вторых, мы представляем алгоритм с разделением параметров между агентами с градиентным спуском Сарса (λ лямбда) (PS-MAGDS) для обучения наших юнитов. Используя нейронную сеть в качестве аппроксиматора функций, агенты обмениваются параметрами централизованной политики и одновременно обновляют политику в соответствии с собственным опытом. Этот метод эффективно обучает однородных агентов и поощряет командное поведение. Чтобы решить проблему редких и отсроченных вознаграждений, мы вводим функцию вознаграждения, включающую небольшие промежуточные вознаграждения в RL-модели. Эта функция вознаграждения улучшает процесс обучения и служит внутренней мотивацией, которая помогает юнитам сотрудничать друг с другом. В-третьих, мы предлагаем метод трансферного обучения, чтобы расширить нашу модель для различных сценариев. По сравнению с обучением с нуля этот метод ускоряет процесс обучения и в значительной степени повышает эффективность обучения. В крупномасштабных сценариях мы применяем метод постепенного трансферного обучения, чтобы успешно обучать группу юнитов. Что касается процента побед, предлагаемый нами метод превосходит некоторые базовые методы в целевых сценариях.

Оставшаяся часть статьи организована следующим образом. В разделе II мы описываем формулировку проблемы микроуправления в StarCraft, а также основы обучения с подкреплением и постепенное трансферное обучение. В Разделе III мы представляем модель обучения с подкреплением для микроуправления, включая метод представления состояния, архитектуру нейросети и определение действий. А в Разделе IV мы представляем алгоритм с разделением параметром между агентами с градиентным спуском Сарса(λ) и функцию вознаграждения. В разделе V мы представляем сценарии микроуправления StarCraft, используемые в нашей статье, и подробности обучения. В разделе VI мы проводим анализ результатов эксперимента и обсуждаем изученные стратегии. В конце мы подведем итоги работы и предложим некоторые будущие работы.

# 2. Постановка проблемы и предпосылки

## 2-a. Постановка проблемы

В микроуправлении StarCraft нам нужно управлять группой юнитов, чтобы уничтожать врагов при определенных условиях местности. Боевой сценарий с несколькими юнитами аппроксимируется как игра Маркова, мультиагентным расширением марковских процессов принятия решений (MDP) [21], [22], [23]. В марковской игре с N агентами, набор состояний S используется для описания свойств всех агентов и среды, а также набора действий A1, ..., AN и наблюдений O1, ..., O N для каждого агента.

В бою юниты c каждой стороны должны кооперироваться друг с другом. Разработка модели обучения для нескольких юнитов является сложной задачей микроуправления. Чтобы поддерживать гибкую структуру и допускать произвольное количество юнитов, мы считаем, что наши юниты должны иметь доступ к пространству состояний S из собственного наблюдения за текущим боем, рассматривая других юнитов как часть среды S → O i. Каждый юнит взаимодействует в боевой обстановке исходя из своих наблюдений и действий. S × A 1 × ... × A N → S 0 обозначает переход из состояния S в следуещее состояние S0 с действиями всех подразделений, а R1 ... RN - сгенерированные вознаграждения каждого юнита. Что касается много-агентной кооперации, политика распространяется среди наших юнитов. Цель каждого юнита - максимизировать его общий ожидаемый результат.

## 2-b. Обучение с подкреплением

Чтобы решить проблему мультиагентного управления в микроменджменте StarCraft, мы можем прибегнуть к обучению с подкреплением. Обучение с подкреплением - это тип алгоритмов машинного обучения, в котором агенты учатся методом проб и ошибок и определяют идеальное поведение на основе своего собственного опыта работы с окружающей средой [24]. Мы рисуем классическую RL-диаграмму на рис. 1. Она показывает процесс взаимодействия RL-агента с окружающей средой. Процесс взаимодействия агент-среда в RL сформулирован как процесс принятия решений Маркова. Агент в состоянии s выполняет действие A согласно политике π.

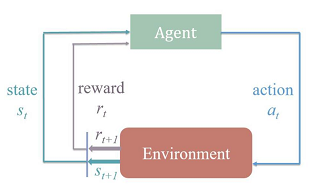


Рис. 1. Представление взаимодействия агентов со средой в обучении с подкреплением

Такое поведение вызывает вознаграждение r и переводит в новое состояние s0. Мы определяем будущий дисконтированный результат в момент времени t как , где T - конечный временной шаг, а γ ∈ [0, 1](гамма) - коэффициент дисконтирования, который определяет важность будущих вознаграждений. Цель RL-модели состоит в том, чтобы изучить оптимальную политику π, которая определяет вероятность выбора действия a в состоянии s, чтобы сумма всех дисконтированных вознаграждений была максимизирована, как показано

(1)

Как один из самых популярных алгоритмов RL, обучение с разницей во времени (TD) представляет собой комбинацию метода Монте-Карло и метода динамического программирования. Метод TD может учиться на основе необработанного опыта без модели среды и обновлять оценки на основе части последовательности, не дожидаясь окончательного результата [25]. Наиболее широко известными алгоритмами обучения TD являются Q-learning и Sarsa. Q-learning оценивает ценность совершения действия в данном состоянии и итеративно обновляет оценку Q-значения в соответствии с наблюдаемым вознаграждением. Ошибка TD δ(дельта) t в Q-обучении вычисляется как

(2)

Q-learning - это метод обучения вне политики, что означает, что он изучает различные политики сравнивая с одним выбранным действем. В отличие от механизма вне-политичного Q-learning, Sarsa - это метод, основанный на политике, что означает, что политика используется как для выбора действий, так и для обновления предыдущего значения Q [24]. Правило обновления Sarsa демонстрируется как



где α - скорость обучения. Традиционные методы обучения с подкреплением имеют несколько успешных применений, в том числе TD в нардах [26] и адаптивное динамическое программирование (ADP) в управлении [27], [28], [29].

В последние годы большое внимание уделяется обучению с подкреплением с глубокой нейронно-сетевой(deep neural networks, DRL) функцией апроксиматора. DRL предоставляет возможность обучить агентов для решения ряда задач на уровне человека непосредственным способом [30] [31]. Как самый известный алгоритм DRL, глубокая Q-сеть (DQN) использует технику воспроизведения опыта и целевую сеть для удаления корреляций между выборками и стабилизации процесса обучения [5]. За последние несколько лет мы стали свидетелями большого числа улучшений DQN, включая двойной DQN [32], приоритетной DQN [33], дуэльной DQN [34], распределенной DQN [35] и асинхронной DQN [36].

Помимо основанных на значениях методов DRL, таких как DQN и его вариантов, методы DRL на основе политики используют глубокие сети для параметризации и прямой оптимизации политики [37]. Глубокий детерминированный градиент политики (Deep deterministic policy gradient, DDPG) - это доработанный аналог DQN, который использует критику для оценки ценности текущей политики и агента для обновления политики [38]. Методы DRL на основе политик играют важную роль в непрерывном контроле, включая асинхронно преимущественную критику агента (asynchronous advantage actor-critic (A3C)) [36], оптимизацию политики в области доверия (trust region policy optimization , TRPO) [39], оптимизацию проксимальной политики (proximal policy optimization, PPO) [40] и так далее.

Образец сложности традиционных методов DRL, как правило, высок, что ограничивает эти методы для реальных приложений. В то время как основанные на модели подходы DRL изучают значение функции и политику эффективных с точки зрения данных способов, они широко используются в сенсомоторном управлении. Поиск руководствующих политик (Guided policy search, GPS) использует контролирующий алгоритм обучения для тренировки политики и алгоритм RL генерирует направляющие распределения, что позволяет эффективно обучать глубоким политикам [41]. Исследователи также предложили некоторые другие основанные на модели методы DRL, такие как нормализованные функции преимущества (normalized advantage functions, NAF) [42] и встраивание в контроль (embed to control, E2C) [43].

Многоагентное обучение с подкреплением тесно связанно с нашей областью работы[44]. Многоагентная система включает несколько агентов, взаимодействующих в одной среде [45] [46]. Недавно были предложены некоторые алгоритмы обучения с многоагентным подкреплением с глубокой нейронной сетью для обучения коммуникации [47], кооперативно-конкурентному поведению [23] и несовершенной информации [48]. В нашей работе мы используем многоагентный метод обучения с подкреплением с политикой разделения между агентами для изучения кооперативного поведения. Агенты разделяют параметры централизованной политики и одновременно обновляют политику в соответствии со своим опытом. Этот метод может обучать гомогенных агентов более эффективно [49].

## 2-c. Постепенное трансфертное обучение

Вообще говоря, безмодельные методы обучения с подкреплением нуждаются в большом количестве примеров, чтобы изучить оптимальную политику. Тем не менее, многие сложные задачи сложны для традиционных RL-методов, чтобы изучить допустимые политики в большом пространстве состояний и действий. В микроуправлении StarCraft существует множество сценариев с различными юнитами и условиями местности.

Это займет много времени, чтобы изучить полезные стратегии в различных сценариях с нуля. Ряд исследователей сосредоточены на повышении скорости и производительности обучения, используя знания предметной области для решения различных, но связанных задач. Наиболее широко используемым подходом является трансферное обучение (TL) [50] [51]. В некоторой степени, трансферное обучение является своего рода обобщением между задачами, передачей знаний от исходных задач к целевым задачам. Кроме того, трансферное обучение может быть распространено на RL-задачи с использованием параметров модели в той же архитектуре модели [52]. Процедура использования трансферного обучения в наших экспериментах - это сначала обучение модели RL-методом в исходном сценарии. И затем мы можем использовать хорошо обученную модель в качестве отправной точки для изучения микроуправления в целевых сценариях.

Как особая форма трансферного обучения, обучение по плану включает в себя набор задач, организованных в соответствии с возрастающим уровнем сложности. Начальные задания используются для того, чтобы направлять учащегося, чтобы он смог лучше выполнять заключительное задание [53]. Комбинируя обучение по плану и трансферное обучение, метод плановоого трансферного обучения (CTL) показал хорошие результаты, помогая процессу обучения быстрее сходиться и достигать оптимума в недавних работах[54], [55], [56]. Для микроуправления, наиболее удобный метод использования CTL - это сначала освоение простого сценария, а затем решение сложных сценариев на основе этих знаний. Изменяя количество и тип юнитов, мы можем контролировать сложность микроуправления. Таким образом, мы можем использовать CTL для обучения наших юнитов с последовательностью постепенно усложняющихся сценариев микроуправления, как показано на рис. 2.

# 3. Обучение модели микроменеджменту

## 3-a. Представление многомерного состояния

Представление состояний в StarCraft остается открытой проблемой без универсального решения. Мы строим представление состояний с помощью входных данных из игрового движка, которые имеют разные типы данных и измерения, как показано в таблице I. Предложенный метод представления состояния эффективен и не зависит от количества юнитов в бою. Таким образом, представление состояния состоит из трех частей: информация о состоянии текущего шага, информация о состоянии последнего шага и действия последнего шага, как показано на рис. 3.

Информация о состоянии текущем шага включает в себя время перезарядки собственного оружия, количество очков здоровья, информацию о расстояниях до своих юнитов, информацию о расстоянии до вражеских юнитов и информацию о расстоянии до объектов местности. Информация о состоянии последнего шага совпадает с текущим шагом. Мы принимаем к сведению последнее действие, которое оказалось полезным для процесса обучения в области RL [57], [58]. Предложенный метод представления состояния также имеет хорошее обобщение и может использоваться в других боевых играх, в которых необходимо учитывать свойства юнитов и информацию о расстоянии между ними.

Все входы с вещественным типом нормированы по их максимальным значениям. Среди них CoolDown и HitPoint имеют 1 измерение для каждого. Мы делим боевую карту на 8 секторов в среднем и вычисляем информацию о расстояниях в каждой области. Информация о расстоянии между единицами указана следующим образом:

* OwnSumInfo: расстояния между собственными юнитами суммируются в каждой области;
* OwnMaxInfo: расстояния между собственными юнитами максимизированы в каждой области;
* EnemySumInfo: расстояния между вражескими юнитами суммируются в каждой области;
* EnemyMaxInfo: расстояния между вражескими юнитами максимизированы в каждой области.

Если юнит находится вне зоны видимости центрального юнита D, значение расстояния юнита устанавливается на 0,05. В противном случае значение линеаризутеся с d, расстояние до центрального юнита, как показано в уравнении (4). Кроме того, значение расстояния между объектами местности рассчитывается также в 8 секторах. Если препятствие находится вне зоны видимости центрального юнита, значение устанавливается равным 0. В противном случае значение также является линеаризуется с расстоянием до центрального юнита, как показано в уравнении (5).

Таким образом, информация о состоянии текущего шага имеет 42 измерения. Действие последнего шага имеет 9 измерений, причем для выбранного действия задано значение 1, а для других действий - 0. В общем, в представление состояния в нашей модели встроено 93 измерения

## 3-b. Определение действий

В сценариях микроуправления StarCraft исходное пространство действий очень велико. На каждом временном шаге каждый юнит может перемещаться в произвольных направлениях с произвольными расстояниями на карте. Когда юнит решает атаковать, он может выбирать любого вражеского юнита в радиусе действия оружия. Чтобы упростить пространство действий, мы выбираем 8 направлений движения с фиксированным расстоянием и выбираем действие «атакуем самых слабых» в качестве доступного действия для каждого юнита.

Когда выбранное действие - движение, наши юниты повернутся в одно из 8 направлений: вверх, вниз, влево, вправо, влево вверх, право вверх, влево вниз, право вниз и переместятся на фиксированное расстояние. Когда выбранное действие является атакой, наши юниты останутся на текущей позиции и будут фокусировать огонь на вражеских юнитах. В настоящее время в качестве цели мы выбираем противника с самым низким количеством здоровья в радиусе атаки нашего оружия. Согласно результатам эксперимента, этих действий достаточно, чтобы контролировать наших юнитов в игре.

## 3-c. Архитектура нейросети

Поскольку опыт наших юнитов является ограниченным подмножеством большого пространства состояний, и большинство тестовых состояний никогда не будет исследована заранее, будет трудно применить таблицу обучение с подкреплением для изучения оптимальной политики. Чтобы решить эту проблему, мы используем нейронную сеть, параметризованную вектором θ(тета), для аппроксимации значений действия состояния для улучшения обобщения нашей RL-модели. Ввод сети - тензор с 93 измерениями для представления состояния. У нас есть 100 нейронов в скрытом слое, и мы используем функцию усечённую линейную функцию(ReLU) для нелинейности сети, что демонстрируется

f(z) = max (0, z), (6)

где z - выход скрытого слоя. Мы используем функцию ReLU, а не функцию Sigmoid или tanh, потому что функция ReLU не имеет проблемы градиентного спуска, что может гарантировать эффективное обучение модели [59]. В отличие от этих насыщающенных функций нелинейности, таких как сигмоид или тан, функция ReLU является ненасыщенной нелинейной функцией. С точки зрения времени обучения с градиентным спуском ненасыщенная нелинейная намного быстрее [60]. Выходной слой нейронной сети имеет 9 нейронов, что дает вероятность движения в 8 направлениях и атаки. Модель обучения одного юнита в сценариях микроуправления StarCraft, включает представление состояния, архитектуру нейронной сети и выходные действия, изображена на рис. 3.

# 4. Метод обучения микроменеджменту

В этой статье мы формулируем микроуправление StarCraft как многоагентную модель обучения с подкреплением. Мы предлагаем метод многоагентного градиентно-спуска Сарса (λ) (PS-MAGDS) с разделением параметров для обучения модели, и конструируем функцию вознаграждения в качестве внутренней мотивации для продвижения процесса обучения. Вся схема PS-MAGDS обучения с подкреплением изображена на рис. 4.

## 4-a. Много-агентный алгоритм с градиентным спуском и разделением параметров Сарса(λ)

Мы предлагаем RL-алгоритм с несколькими агентами, который расширяет традиционный Сарса алгоритм (λ) для управления нескольких юнитов, разделяя параметры сети политики между нашими юнитами. Чтобы ускорить процесс обучения и решить проблему отсроченных вознаграждений, мы используем приемлемые черты(eligibility traces) в обучении с подкреплением. В качестве основного механизма в RL, приемлемые черты используются для назначения временного вознаграждения, который учитывает набор ранее испытанных переходов [61]. Это означает, что он учитывает не только значение последней пары состояние-действие, но и посещенные. С помощью этого метода мы можем решить проблему отсроченного вознаграждения в игровой среде. Сарса с приемлемыми чертами, называемая Сарса (λ лямбда), является одним из способов усреднения резервных копий(backups), сделанных после нескольких шагов. λ является фактором, который определяет вес каждой резервной копии. В нашей реализации Sarsa (λ) для боя нескольких юнитов, мы используем нейронную сеть в качестве аппроксиматора функции и разделяем параметры сети между всеми нашими юнитами. Хотя у нас есть только одна сеть для обучения, подразделения могут вести себя по-разному, потому что каждый юнит получает разные наблюдения и действия в качестве входных данных.

Для эффективного обновления сети политик мы используем метод градиентного спуска для обучения модели обучения с подкреплением Сарса (λ). Обновление модели с градиентным спуском продемонстрировано в уравнении (7),

δt = rt+1 + γQ(st+1, at+1; θt) − Q(st, at; θt) (7a)

θt+1 = θt + αδtet (7b)

et = γλet−1 + ∇θtQ(st, at; θt), e0 = 0 (7c)

где et приемлемые черты на шаге t.

Одним из сложных вопросов в обучении с подкреплением является компромисс между разведкой и эксплуатацией. Если мы выберем лучшее действие на каждом этапе в соответствии с текущей политикой, мы, вероятно, попадем в локальный оптимум. Напротив, если мы склонны исследовать в большом пространстве состояний, модели будет трудно сходиться. В эксперименте мы используем метод e-жадный для выбора действий во время тренировки, который выбирает текущее наилучшее действие с вероятностью 1 - e и выполняет случайное исследовательское действие с вероятностью e,

(8)

где N равно 9 в эксперименте.

Мы используем экспоненциально e распад для реализации метода e-жадный. e инициализируется со значением 0,5 и график закалки с показателем экспоненциального сглаживания окна в эпизоде с номером эпизода *episode\_num*, как показано

(9)

Целиком метод разделения общих параметров мультиагентного градиентного спуска Сарса (λ) представлен в алгоритме 1.

## 4-b. Функция вознаграждения

Функция вознаграждения обеспечивает полезную обратную связь для RL-агентов, которая оказывает значительное влияние на результаты обучения [62]. Целью микроуправления StarCraft является уничтожение всех вражеских .юнитов в бою. Если вознаграждение основано только на конечном результате, функция вознаграждения будет крайне скудной. Более того, юниты обычно получают положительное вознаграждение после множества шагов. Отсрачивание вознаграждения затрудняет выяснение того, какой набор действий отвечает за соответствующие вознаграждения.

Чтобы решить проблему редких и отсроченных вознаграждений в микроуправлении, мы разработали функцию вознаграждения, включающую небольшие промежуточные вознаграждения. В нашем эксперименте все агенты получают основное вознаграждение, получаемое в случае атаки на каждом временном шаге, равное урону, который получили вражеские юниты минус количество потерянного здоровья наших юнитов.

*rt = (damage\_amountt × damage\_factor − ρ×(unit hitpointt−1 − unit hitpointt))/10* (10)

где *damage\_amount* - это количество урона, нанесенного атакой наших юнитов, *damage\_factor* - сила атаки наших юнитов, а количество *хитпоинтов* юнита — это количество очков здоровья нашего юнита. Мы делим вознаграждение на константу, чтобы изменить коэффициент до более подходящего диапазона, который в нашем эксперименте установлен на 10. ρ является нормализованным фактором для баланса общего количества очков здоровья наших юнитов и юнитов противника,



где H - количество вражеских юнитов, а N - количество наших юнитов. Вообще говоря, этот нормализованный фактор необходим в микроуправлении StarCraft с различным количеством и типом юнитов. Без надлежащей нормализации, политика сети будет хуже сходиться, и нашим юнитам нужно гораздо больше времени для изучения полезного поведения.

Помимо основной награды за атаку, мы рассматриваем некоторые дополнительные награды как внутреннюю мотивацию для ускорения тренировочного процесса. Когда юнит уничтожается, мы вводим дополнительное отрицательное вознаграждение и присваеваем значение -10 в нашем эксперименте. Мы хотели бы наказать такое поведение, потому-что уменьшение количества собственных юнитов плохо влияет на результат боя. Кроме того, чтобы побудить наших юнитов работать в команде и предпринять совместные действия, мы вводим вознаграждение за их перемещение. Если в направлении движения нет наших юнитов или вражеских юнитов, мы даем этому движению небольшое отрицательное вознаграждение, которое равно -0,5. Согласно эксперименту, это вознаграждение оказывает впечатляющее влияние на эффективность обучения, как показано на рис. 6.

## 4-c. Пропуск кадров

Применяя обучающее подкрепление к видеоиграм, мы должны обратить внимание на непрерывность действий. Так как микроуправление в StarCraft происходит в реальном то нецелесообразно совершать действия в каждом игровом кадре. Одним из возможных способов является использование технологии пропуска кадров, которая выполняет этап обучения для каждого фиксированного числа кадров. Тем не менее, небольшой пропуск кадров привнесет сильную корреляцию в обучающие данные, в то время как большой пропуск кадров уменьшит эффективные обучающие выборки. Мы ссылаемся на соответствующую работу в [20] и пробуем несколько пропусков кадров (8, 10, 12) в небольшом сценарии микроуправления. Наконец, мы установили пропуск кадра равным 10 в нашем эксперименте, который выполняет действие каждые 10 кадров для каждого юнита.

# 5. Постановка эксперимента

## 5-a. Сценарии микроуправления в Starcraft

Мы рассматриваем несколько сценариев микроуправления StarCraft с различными юнитами, включая Голиафы против Зилотов, Голиафы против Зерглингов и Морпехов против Зерглингов, как показано на Рис. 5.

1) В первом сценарии мы будем управлять тремя Голиафами, чтобы сражаться с 6 Зилотами. Из таблицы II видно, что вражеские юниты имеют преимущество по количеству юнитов, здоровья и коэффициенту урона. В отличие от них, радиус атаки наших юнитов намного шире.

2) Во втором сценарии у врагов есть 20 зерглингов. Наши Голиафы имеют преимущество по количеству здоровья, коэффициенту урона и дальности огня, в то время как у врагов гораздо больше юнитов и меньше времени перезарядки(то есть выше скорость атаки).

3) В третьем сценарии мы будем управлять 20-ю морпехами для борьбы с 30 зерглингами. Вражеские юниты имеют преимущество в скорости и количестве, в то время как наши юниты имеют преимущество в дальности огня и коэффициенте повреждения.

Мы делим эти сценарии на две группы. Первый и второй являются мелкомасштабными микроуправлениями, а последний - крупномасштабным микроуправлениям. В этих сценариях вражеские юниты управляются встроенным ИИ, который жестко запрограммирован в игре. Эпизод заканчивается, когда любая из сторон потеряла всех своих юнитов. Человек-новичок в StarCraft не может победить встроенный ИИ в этих сценариях. Игроки с платиновым уровнем имеют средний процент выигрыша ниже 50% и 100 игр для каждого сценария. Ожидается, что наши RL-агенты будут использовать их преимущества и избегать их недостатков, чтобы выиграть эти бои.

## 5-b. Обучение нейросети

В процессе обучения мы устанавливаем коэффициент дисконтирования γ равным 0,9, коэффициент обучения α равным 0,001, а коэффициент приемлимых черт λ - 0,8 во всех сценариях. Кроме того, максимальное количество шагов в каждом эпизоде равен 1000. Чтобы ускорить процесс обучения, игра запускается на полной скорости, установив gameSpeed в 0 в BWAPI. Эксперимент проводится на компьютере с процессором Intel i7-6700 и 16 ГБ памяти.

# 6. Результаты и обсуждения

В этом разделе мы анализируем результаты различных сценариев микроуправления и обсуждаем производительность нашей RL-модели. В небольших сценариях мы используем первый сценарий в качестве отправной точки для обучения наших юнитов. В оставшихся сценариях мы вводим трасферный метод обучения, чтобы масштабировать бои до больших сценариев. Цель микроуправления StarCraft - победить врагов и увеличить выигрыш в данных сценариях. Для лучшего понимания мы анализируем коэффициенты выигрыша, количество этапов и среднее вознаграждение во время тренировки, а также изученные стратегии. Наш код и результаты с открытым исходным кодом на https://github.com/nanxintin/StarCraft-AI.

## 6-a. Микроуправление малого масштаба

В мелкомасштабных сценариях микроуправления мы будем тренировать Голиафов против вражеских юнитов с разным количеством и типом. Во втором сценарии мы также будем использовать метод трансферного обучения для обучения Голиафов на основе хорошо обученной модели первого сценария. Оба сценария обучены на 4000 эпизодах и более 1 миллиона шагов.

1) Голиафы против зилотов: в этом сценарии мы обучаем отряд наших Голиафов с нуля и анализируем результаты.

* Процент побед: Сначала мы проанализируем эффективность обучения нашего метода RL с помощью moveReward. Чтобы оценить количество побед, мы тестируем нашу модель после каждых 200 эпизодов тренировок для 100 боев и показываем результаты на рис. 6. Мы видим, что наши отряды Голиафа не могут выиграть ни одного боя до 1400 эпизодов. С прогрессом обучения юниты начинают выигрывать в нескольких играх, а кривая показателей выигрыша впечатляет после 2000 эпизодов. После 3000 эпизодических тренировок наши отряды могут наконец достичь 100% выигрыша.
* Шаги эпизодов: мы показываем средние количество шагов в эпизоде и стандартные отклонения наших трех Голиафов во время обучения на Рис. 7. Очевидно, что кривая средних шагов эпизода имеет четыре стадии. В начале, эпизодов очень мало, потому что Голиафы ничему не научились и быстро уничтожаются. После этого Голиафы начинают понимать, что урон здоровью вызывает отрицательное вознаграждение. Они учатся убегать от врагов, и количество шагов в эпизоде увеличиваются до высокого уровня. И затем, шаги эпизода начинают уменьшаться, потому что Голиафы учатся атаковать, чтобы получить положительные награды, а не просто убегать. В конце концов, Голиафы выучили соответствующую политику, чтобы уравновесить маневрённость и атаку, и они способны уничтожать врагов почти за 300 шагов.
* Среднее вознаграждение: Вообще говоря, мощный игровой ИИ в сценариях микроуправления должен побеждать врагов как можно скорее. Здесь мы вводим среднее вознаграждение, деля общее вознаграждение на шаги эпизода в бою. Кривая среднего вознаграждения наших Голиафов изображена на рис. 8. Средние вознаграждения имеют очевидное увеличение при открытии, и неуклонно растёт во время тренировок и остаются плавными после почти 3000 эпизодов.

2) Голиафы против зерглингов: в этом сценарии вражеские отряды представляют собой группу зерглингов, и мы повторно используем хорошо обученную модель из первого сценария для инициализации сети политик. По сравнению с обучением с нуля, у нас есть лучшее понимание трансферного обучения.

Коэффициенты выигрыша. На рис. 9 показаны коэффициенты выигрыша. При обучении с нуля процесс обучения очень медленный, и наши юниты не могут выиграть игру до 1800 эпизодов. Без трансферного обучения процент выигрышей ниже 4000% после 4000 эпизодов. Когда обучение основано на модели первого сценария, процесс обучения происходит намного быстрее. Даже в дебюте наши юниты выигрывают несколько игр, и в итоге процент выигрышей достигает 100%.

Шаги эпизода: На рис. 10 мы показываем средние количество шагов эпизода для наших трех Голиафов во время тренировки. Без трансферного обучения кривая имеет сходную тенденцию с таковой в первом сценарии. Средние количество шагов эпизода имеет очевидное увеличение открытия и постепенно синжается во время тренировки. При обучении с трансферным обучением к среднее количество шагов эпизода остаётся стабильными в течение всего тренировочного процесса, в диапазоне от 200 до 400. Возможное объяснение состоит в том, что наши юниты изучили некоторые базовые навыки движения и атаки из хорошо обученной модели, и они используюте эти навыки, чтобы ускорить процесс обучения.

Средняя награда. На рис. 11. мы показываем среднюю награду для наших трех Голиафов. При обучении с нуля наши юниты испытывают трудности в победе в бою в дебюте, а средние награды находятся на низком уровне до 1000 эпизодов. Средние вознаграждения с трансферным обучением, для сравнения, намного выше с самого начала и ведут себя лучше во всем процессе обучения.

В крупномасштабных сценариях микроуправления мы используем постепенное трансферное обучение, чтобы обучить наших морпехов играть против зерглингов и сравнить результаты с некоторыми базовыми методами.

1) Морпехи против зерглингов: в этом разделе мы разрабатываем учебную программу с 3 классами для обучения юнитов, как показано в таблице III. После обучения мы тестируем производительность в двух целевых сценариях: M10 против Z13 и M20 против Z30. Кроме того, мы используем некоторые базовые методы для сравнения, которые состоят из подходов основанных на правилах и подходов DRL.

* Слабейшие: метод, основанный на правиле, атаковать слабейших в радиусе действия оружия.
* Ближайший: метод, основанный на правиле, атакуовать ближайшего в радиусе действия оружия.
* GMEZO: метод DRL, основанный на оптимизации нулевого порядка, имеющий впечатляющие результаты по сравнению с традиционными RL-методами [20].
* BicNet: метод DRL, основанный на архитектуре актер-критика, имеющий наилучшую производительность в большинстве сценариев микроуправления StarCraft [21].

В таблице IV мы представляем процент выигрышей по методу PS-MAGDS и базовым методам. В каждом сценарии мы 5 раз измеряем средние показатели выигрышей в нашей модели в 100 тестовых играх. В M10 против Z13 PS-MAGDS достигает выигрыша 97%, что намного выше, чем другие методы, включая недавно предложенные GMEZO и BicNet. В M20 против Z30, PS-MAGDS имеет вторую лучшую производительность, которая очень близка к лучшей.

Мы также проверяем наши хорошо обученные модели в учебных сценариях и не исследуемых ранее сценариях, и представляем результаты в Таблице V. Мы видим, что у PS-MAGDS есть выдающиеся производительность в этих учебных сценариях. В не исследованных ранее сценариях с большим количеством единиц у PS-MAGDS также есть приемлемые результаты.

## 6-c. Анализ стратегий

В микроуправлении StarCraft есть различные типы юнитов с различными навыками и свойствами. Игроки должны изучить, как двигаются и атакуют группы юнитов в режиме реального времени. Если мы проектируем основанный на правилах AI, чтобы решить эту проблему, мы должны рассмотреть большое количество условий, и способность агентов также ограничена.

Новички StarCraft не могут выиграть ни один из этих боев, представленных в нашей статье. Таким образом, это поведение очень сложное и трудно в усвоениии. При обучение с подкреплением и постепенным обучением, наши юниты в состоянии справиться с несколькими полезными стратегиями в этих сценариях. В этом разделе мы сделаем краткий анализ этих стратегий, которые изучили наши юниты.

1) Разделение врагов: В мелкомасштабных сценариях микроуправления наши Голиафы должны бороться против противника с большим количеством противников и большим количеством общего количества здоровья. Если наши отделения останутся вместе и будут бороться с противником «лицом к лицу», они будут быстро уничтожены и проиграют бой. Правильная стратегия в разделении врагов и уничтожении их по одному.

В первом сценарии наши отделения Голиафов изучили рассеивающихся зилотов после обучения. В открытии наши отделения рассеивают врагов в несколько частей и разрушают его в одной части сначала. После этого победа Голиаф двигается к другим Голиафам и помогает бороться против врагов. Наконец, наш центр единиц стреляют в остающихся врагов и разрушают их. Для лучшего понимания мы выбираем некоторые структуры переигровки игры в бою и привлекаем направления движения и нападения единиц на Рис. 12. Белые линии обозначают направления движения, и красные линии обозначают направления нападения.

Подобная стратегия происходит во втором сценарии. У противника есть намного больше единиц, и ранняя атака Зерглингами наносит большой ущерб противнику, она часто используется в StarCraft. Наши Голиафы разделяют Зерглингов на несколько групп и сохраняют подходящую дистанцию. Когда оружие юнита находится в состоянии готовым к стрельбе, они прекращают двигаться и нападают на врагов, как показано на Рис. 13.

2) Сохранение команды. В масштабных сценариях микроуправления каждая сторона имеет массу юнитов. Морпехи - это небольшие наземные подразделения с низкими количеством здоровья. Если они сражаются в нескольких небольших группах, они не могут противостоять врагам. Подходящей стратегией является удержание наших морпехов в команде, движение в том же направлении и атака на одну и ту же цель, как показано на рис. 14. Из этих рисунков мы видим, что наши морпехи научились двигаться вперед и отступать по очереди.

3) Бей и беги: кроме глобальных стратегий, о которых говорилось выше, наши юниты также изучили некоторые локальные стратегии во время обучения. Среди них «Бей и беги» - наиболее широко используемая тактика в микроуправлении StarCraft. Наши юниты быстро изучают тактику «Беги и беги» во всех сценариях, включая «Бей и беги» одного юнита на рис. 12 и рис. 13, а также группу юнитов использующих тактику «Бей и беги»на рис. 14

4) Существующие проблемы: Хотя наши подразделения выучили полезные стратегии после обучения, в сражениях все еще есть некоторые проблемы. Например, Голиафы время от времени двигаются вперед и назад и не участвуют в битвах, чтобы вовремя помогать другим юнитам. Кроме того, юниты предпочитают перемещаться к границе карты, чтобы избежать врагов.

# 7. Заключение и будущие работы

Эта статья посвящена управлению несколькими юнитами в сценариях микроуправления StarCraft. Мы представляем несколько вкладов, в том числе эффективное представление состояния, совместное использование параметров многоагентного градиентного спуска Сарса (λ), эффективную функцию вознаграждения и метод трансферного постепенного обучения, используемый для расширения нашей модели в различных сценариях.

Мы демонстрируем эффективность нашего подхода как в мелкомасштабных, так и в крупномасштабных сценариях, а также превосходную производительность по сравнению с некоторыми базовыми методами в двух целевых сценариях. Примечательно, что предлагаемый нами метод способен выучить соответствующие стратегии и победить встроенный ИИ в различных сценариях. Кроме того, еще есть области для будущей работы.

Совместное поведение нескольких юнитов изучается путем совместного использования сети политик, построения эффективного метода представления состояния, включающего в себя информацию других юнитов и предлагаемую встроенную функцию мотивированного вознаграждения. Хотя наши подразделения могут успешно освоить некоторые эффективные стратегии координации, мы будем исследовать более интеллектуальные методы для совместной работы нескольких агентов. Чтобы решить проблему отложенного вознаграждения в микроуправлении StarCraft, мы используем простой, прямой и эффективный метод формирования вознаграждения. Тем не менее, есть также некоторые другие методы решения редких и отсроченных вознаграждений, такие как иерархическое обучение с подкреплением. Иерархический RL объединяет иерархические функции действия-значения, работающие в разных временных масштабах [63]. По сравнению с методом формирования вознаграждения, иерархический RL обладает способностью изучать абстрагированное от времени исследование, и дает агентам большую гибкость.

Но его структура также намного сложнее, и автоматическое извлечение подзадач остается открытой проблемой. В будущем мы проведем углубленное исследование по применению иерархической RL к StarCraft. В настоящее время мы можем обучать только наземных юнитов дальнего боя того же типа, в то время как тренировка наземных юнитов ближнего боя с использованием методов RL остается открытой проблемой. Мы улучшим наш метод для большего количества типов юнитов и более сложных сценариев в будущем. Наконец, мы также рассмотрим возможность использования нашей модели микроуправления для StarCraft бота в полноценной игре.

# Приложения

## Терминологический словарь

Обучение с подкреплением(reinforcement learning) – один из методов машинного обучения, в ходе которого испытуемая система (агент) обучается, взаимодействуя с некоторой средой.

Трансферное обучение(transfer learning) - это метод машинного обучения, в котором нейросеть, разработанная для одной задачи, повторно используется в качестве отправной точки для другой нейросети, которая решает схожую задачу.

Постепенное обучение(или обучение по плану)(curriculum learning) - обучение нейросети определённой последовательности постепенно усложняющихся задач, которые помогут в достижении конечной цели.

Марковский процесс принятия решений (англ. Markov decision process (MDP)) — спецификация задачи последовательного принятия решений для полностью наблюдаемой среды с марковской моделью перехода и дополнительными вознаграждениями.

Це́пь Ма́ркова — последовательность случайных событий с конечным или счётным числом исходов, характеризующаяся тем, что действия могут совершаться независимо от прошлых действий.

Игра Маркова – игра, где последовательность действий игрока(ов) будет цепью Маркова.