ДОКЛАДЫ РОССИЙСКОЙ АКАДЕМИИ НАУК. МАТЕМАТИКА, ИНФОРМАТИКА, ПРОЦЕССЫ УПРАВЛЕНИЯ, 2023, том 514, № 2, с. 250–261

УДК 004.8

ИЕРАРХИЧЕСКИЙ МЕТОД КООПЕРАТИВНОГО МУЛЬТИАГЕНТНОГО ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ В МАРКОВСКИХ ПРОЦЕССАХ ПРИНЯТИЯ РЕШЕНИЙ

© 2023 г. В. Э. Большаков[1,*], А. Н. Алфимцев[1,]**

Представлено академиком РАН А.А. Шананиным
Поступило 01.09.2023 г.
После доработки 29.09.2023 г.
Принято к публикации 18.10.2023 г.

В быстро развивающейся области обучения с подкреплением слияние иерархических и мультиагентных методов обучения преподносит уникальные трудности и открывает новые возможности. В данной статье рассматривается сочетание многоуровневого иерархического обучения с обнаружением промежуточных целей и мультиагентного обучения с подкреплением с воспроизведением ретроспективного опыта. Объединение таких подходов приводит к созданию единого метода иерархического мультиагентного обучения с подкреплением, который позволяет множеству агентов эффективно обучаться в сложных средах, в том числе в средах с редкими вознаграждениями. В работе демонстрируются результаты предлагаемого метода в одной из таких сред внутри стратегической игры StarCraft II, и проводится сравнение с другими существующими подходами. Предлагаемый метод разработан в парадигме централизованного обучения с децентрализованным исполнением, что позволяет достичь баланса между координацией и автономностью агентов.

Ключевые слова: мультиагентное обучение с подкреплением, иерархическое обучение, обнаружение промежуточных целей, воспроизведение ретроспективного опыта, централизованное обучение с децентрализованным исполнением, редкие вознаграждения

DOI: 10.31857/S2686954323601501, **EDN:** GQOIZV

1. ВВЕДЕНИЕ

В обучении с подкреплением (англ. Reinforcement Learning, RL) агент взаимодействует со средой и получает от нее сигнал вознаграждения [1]. Действуя методом проб и ошибок, со временем агент учится максимизировать ожидаемое будущее вознаграждение. Подобное вознаграждение называется внешним, получаемым от среды, и, как правило, определяется в соответствии с решаемой задачей, а сама функция вознаграждения задается при помощи экспертных знаний и в явном виде. С помощью такого подхода были получены прорывные результаты машинного обучения в виртуальных мирах компьютерных игр [2–4] и в задачах управления объектами реального мира [5].

В задачах обучения с подкреплением, где задействовано множество одновременно обучающихся агентов, больше не работают теоретические гарантии достижения оптимальной стратегии действий, а применение классических методов одноагентного обучения уже не дает успешных результатов. Агенты делают среду взаимодействия нестационарной, недетерминированной и непредсказуемой, мешая друг другу обучаться [6]. Но при этом ряд задач из реального мира удобно и естественно представляются именно в виде кооперативных мультиагентных систем: управление трафиком, координация автономных устройств, распределение ресурсов [7].

Сегодня одним из самых успешных способов решения проблемы мультиагентного обучения является архитектура исполнитель-критик с использованием схемы централизованного обучения с децентрализованным исполнением (англ. Centralized Training with Decentralized Execution, CTDE) [8]. В ходе обучения централизованный критик получает полную информацию о среде и агентах, что позволяет более точно оценивать действия с учетом влияния агентов друг на друга. Когда обучение закончено, в режиме эксплуатации централизованный критик больше не используется, а агенты выполняют действия самостоятельно, согласно выученным стратегиям.

Другой класс задач, в которых традиционные подходы обучения с подкреплением недостаточно эффективны – это среды с редкими возна-

[1]*Московский государственный технический университет им. Н.Э. Баумана, Москва, Россия*
*****E-mail: bolshakovv@bmstu.ru*
******E-mail: alfim@bmstu.ru*

граждениями [9]. При взаимодействии с такими средами агенту сложнее оценивать успешность своих действий и учиться максимизировать ожидаемое будущее вознаграждение. К примеру, в среде с редкими вознаграждениями агенту часто нужно произвести длинную цепочку действий, прежде чем получить само вознаграждение и понять, насколько его действия были эффективны. Ситуация осложняется тем, что зачастую для исследования среды агент в начале обучения действует случайно, что чрезвычайно понижает его шансы на выполнение успешных длинных цепочек действий.

Иерархическое обучение с подкреплением (англ. Hierarchical Reinforcement Learning, HRL) – это подход к обучению с подкреплением, который включает структуру в процесс принятия решений. Традиционные методы RL часто рассматривают проблемы как одноуровневые, т.е. в них каждое решение в равной степени зависит от предыдущего. Однако многие задачи из реального мира имеют иерархическую структуру, в которой решения могут относиться к задачам высокого уровня и подзадачам более низкого уровня. HRL стремится использовать эту структуру, чтобы сделать обучение более эффективным и масштабируемым.

Подходы к HRL разбивают долгосрочную задачу обучения с подкреплением на иерархию подзадач, так что политика более высокого уровня учится выполнять основную задачу, выбирая оптимальные подзадачи в качестве действий более высокого уровня. Иерархия политик в совокупности определяет поведение агента. Такая декомпозиция эффективно сокращает горизонт планирования исходной задачи до более короткого с точки зрения последовательности подзадач. Теперь каждая подзадача представляет собой действие более высокого уровня, которое выполняется более длительный период времени по сравнению с примитивными действиями, т.е. действиями последнего нижнего уровня иерархии. Это свойство называется временной абстракцией [10–12].

К основным преимуществам иерархических подходов в обучении с подкреплением можно отнести более эффективное назначение вознаграждения в долгосрочных задачах, улучшенное исследование среды за счет того, что подзадачи могут быть относительно быстро выучены и затем переиспользоваться для более эффективного взаимодействия со средой, что также улучшает результаты в задачах с большим пространством состояний, причем агент получает возможность использовать одни и те же выученные навыки или подзадачи для достижения различных целей на соответствующем уровне иерархии. Кроме того, использование иерархии политик улучшает интерпретируемость решений агента, поскольку теперь возможно отследить процесс принятия этих решений на разных уровнях детализации.

В данной статье предлагается иерархический метод для мультиагентного обучения с обнаружением подцелей MASHA (англ. Multi-Agent Subgoal Hierarchy Algorithm), который позволяет добиться эффективного обучения множества агентов в сложной среде с редкими вознаграждениями. При этом в качестве базового алгоритма обучения с подкреплением может использоваться любой нейросетевой алгоритм мультиагентного обучения. Для проведения экспериментов специально разработана оригинальная мультиагентная среда внутри компьютерной стратегической игры StarCraft II, доступ к которой осуществляется посредством интерфейса мультиагентного машинного обучения SMAC [13].

2. ПРЕДЫДУЩИЕ РАБОТЫ

Одним из первых подходов к иерархическому обучению с подкреплением была концепция феодального RL. Идея в том, чтобы повысить эффективность обучения за счет его иерархической структуры, при которой агенты высокого уровня устанавливают абстрактные цели для агентов более низкого уровня. Эти абстрактные цели направляют нижние уровни в исследовании окружающей среды и достижении более крупных целей. Система в этом случае разделяется на управляющего агента и агента исполнителя. Управляющий агент работает в медленном масштабе времени и ставит перед исполнителем подцели, а тот, работая в более быстром масштабе времени, этих подцелей старается достичь. При такой структуре управляющий агент принимает решения на уровне долгосрочных стратегий, а исполнитель занимается немедленными краткосрочными действиями [14].

Феодальные подходы испытывают проблемы из-за нестационарности среды. Поскольку управляющий агент обучается и обновляет свою стратегию, а подцели, которые он ставит исполнителю, могут меняться со временем. С точки зрения исполнителя, такое изменение делает среду нестационарной, поскольку подцели являются частью среды. Помимо этого, текущая оптимальная стратегия исполнителя может перестать быть таковой при получении других подцелей от управляющего агента.

Для решения проблемы нестационарности был разработан метод HIRO [15]. Основной вклад метода HIRO – это способ, гарантирующий, что действия исполнителя будут полезны для управляющего агента, даже если подцели управляющего агента могут быть нестационарными или устаревшими из-за обучения, не связанного с теку-

Таблица 1. Сравнение методов обучения с подкреплением по критериям MAHRL

Метод	Временная абстракция	Масштабируемость иерархии	Мультиагентность	Децентрализованное исполнение
QMIX	–	–	+	+
MADDPG [21]	–	–	+	+
LIIR [22]	–	–	+	+
HIRO	+	–	–	–
HAC	+	+	–	–
HSD	+	–	+	+
MASHA	+	+	+	+

щей стратегией. Изменение меток подцелей в буфере воспроизведения управляющего агента позволяет сделать их более достижимыми и релевантными. Однако метод HIRO был ограничен всего двумя уровнями иерархии.

Независимо от метода HIRO был предложен другой подход, решающий проблему нестационарности в HRL, при этом без ограничения количества уровней иерархии. В методе HAC [16] также использовалась своего рода коррекция меток в буфере воспроизведения, но на основе принципа ретроспективного опыта (Hindsight Experience Replay, HER) [17]. В качестве выходной подцели верхнего уровня и входной подцели нижнего уровня использовалось то состояние среды, в котором оказался агент, пытаясь достичь текущую подцель. Таким образом, каждый уровень иерархии обучался независимо от нижестоящих уровней, считая, что они действуют оптимально и достигают поставленных целей. Вместе с тем метод может применяться только в одноагентном случае машинного обучения с подкреплением.

Другим направлением развития иерархического обучения стали подходы, использующие такие понятия, как навыки или опции. Идея в том, чтобы декомпозировать сложные задачи на простые, более выполнимые подзадачи или навыки. Эти навыки или опции можно изучить автоматически на основе выборки данных или заранее определить на основе экспертных знаний [18].

Исследования проводились и в области мультиагентного HRL (англ. Multi-Agent HRL, MAHRL). Одна из самых заметных работ – метод HSD [19]. В этом методе предлагался способ обнаружения полезных навыков, основанный на вариационном выводе. Для централизованного обучения в парадигме CTDE использовался метод QMIX [20]. Каждый агент использует децентрализованную феодальную иерархию, ограниченную двумя уровнями. Верхний уровень выбирает навык из некоторого пространства, заданного вручную. Нижний уровень использует этот вектор для взаимодействия со средой и генерации примеров поведения, или траекторий движения, а также максимизирует взаимную информацию между различными навыками и порождаемыми траекториями движения, что приводит к исследованию разнообразных навыков.

Рассмотренные методы иерархического обучения с подкреплением приведены в табл. 1, где также указаны и другие подходы, используемые в сравнительных экспериментах в данной работе. В таблице анализируются такие критерии MAHRL, как способность к временной абстракции, т.е. принятию решений на разных масштабах времени, возможность масштабирования иерархии в смысле увеличения количества уровней, возможность обучения множества агентов, а также возможность использования агентов децентрализованно.

3. МОДЕЛЬ HDec-POMDP

Неопределенность, связанная с неспособностью единичного агента получить точное состояние среды, осложняло применение классической модели Марковского процесса принятия решений для обучения с подкреплением. Для преодоления неопределенности состояния среды была разработана математическая модель частично наблюдаемого Марковского процесса принятия решений (POMDP, англ. *Partially Observable Markov Decision Process*), которая на мультиагентный случай расширяется в виде модели децентрализованного POMDP (Dec-POMDP), за счет введения совместных действий и совместных наблюдений агентов [23].

Модель Dec-POMDP представляет собой кортеж: $(N, S, \mathbb{A}, T, R, \mathbb{O}, \Omega, h, b^0)$, где N – множество агентов, S – множество состояний, $\mathbb{A}$ – множе-

ство совместных действий, T – функция переходов, R – функция вознаграждения, $\mathbb{O}$ – множество совместных наблюдений, Ω – функция наблюдений, h – временной горизонт задачи, b^0 – начальное распределение состояний. Множество совместных действий $\mathbb{A}$ состоит из $\mathbb{A} = \times_{i \in n} A_i$, где A_i – множество действий агента i. В каждый момент времени t агент i выполняет некоторое действие a_i, которое приводит к формированию совместного действия мультиагентной системы $a = (a_1, a_2, \ldots, a_n)$. Результат воздействия совместного действия a на среду определяется функцией переходов T как вероятность $T(s'|s,a)$ перейти из состояния s в состояние s'.

Множество совместных наблюдений $\mathbb{O}$ состоит из $\mathbb{O} = \times_{i \in n} \mathbb{O}_i$, где $\mathbb{O}_i$ – множество наблюдений, доступных агенту i. В каждый момент времени t агент i получает некоторое наблюдение o_i, которое образует совместное наблюдение $o = (o_1, o_2, \ldots, o_n)$. Функция наблюдений Ω определяет вероятность восприятия агентами состояния среды как совместного наблюдения $\Omega(o|s',a)$.

Используя модель Dec-POMDP, множество агентов обучается оптимизировать свои действия путем максимизации ожидаемого кумулятивного вознаграждения $R_t = \sum_{t=0}^{h-1} \gamma^{t-1} r_t(o_t, a_t)$ с параметром дисконтирования $\gamma \in [0, 1]$ с некоторым временным горизонтом h с учетом начального распределения состояний среды $b^0 : S \to [0,1]$ в момент времени $t = 0$. При этом поведение агента i определяется стохастической стратегией $\pi_i(a_i|o_i) : \mathbb{O}_i \times A_i \to [0, 1]$. Целью обучения агента i является нахождение оптимальной стохастической стратегии π_i^*, которая максимизирует суммарное вознаграждение агента. Целевая функция агента i может быть задана в виде $J(\theta_i) = \mathbb{E}_{\pi_{\theta_i}}[R_t]$, где θ_i – вектор параметров стохастической стратегии π_i, который в глубоком обучении является весами нейронной сети. Для максимизации целевой функции в процессе машинного обучения параметры стохастической стратегии θ_i изменяются в направлении ее градиента $\nabla_{\theta_i} J(\theta_i)$.

Благодаря теореме о градиенте стратегии [24] было разработано несколько эффективных методов, использующих вычисление градиентов стохастических стратегий, отличающихся только способом нахождения значений функции ценности $Q_{\pi_{\theta_i}}(o_i, a_i)$. Однако стохастическая стратегия $\pi_{\theta_i}(a_i|o_i)$ оставалась зависимой как от множества наблюдений агента $\mathbb{O}_i$, так и от множества действий A_i, но не учитывала наблюдения и действия других агентов, а потому оценки градиента обладали высокой дисперсией. Поэтому была предложена детерминированная стратегия $\mu_{\theta_i}(a_i|o_i) : \mathbb{O}_i \to A_i$, зависящая только от множества наблюдений агента $\mathbb{O}_i$ [25]. В оффлайновом (off-policy) безмодельном методе глубокого детерминированного градиента стратегий DDPG (Deep Deterministic Policy Gradient) стратегия μ_θ и функция ценности Q^{μ_θ} аппроксимировались глубокими нейронными сетями, используя архитектуру исполнитель-критик.

Расширим модель Dec-POMDP для использования в иерархическом подходе. Для этого необходимо включить в модель набор кооперативных целей $\mathcal{G}$ для агентов в мультиагентной системе, из которого в начале каждого эпизода выбирается командная цель $g \in \mathcal{G}$, а стратегия агента i принимает вид $\pi_{\theta_i}(a_i|o_i, g)$. Наконец обозначим иерархию из k уровней как Π_{k-1}. Действиями агентов на последнем, низшем уровне иерархии являются примитивные действия, т.е. атомарные действия внутри среды. Действиями агентов на всех более высоких уровнях являются подцели, такие что пространство подцелей на каждом уровне идентично пространству состояний среды, т.е. $\mathcal{G}_j = S$, где $0 \le j \le k-1$. Наблюдения среды для каждого уровня иерархии и каждого агента выбираются из множества совместных наблюдений среды $\mathbb{O}$. Тогда стратегией агента i на уровне иерархии j является $\pi_{\theta_{i,j}}(a_{i,j}|o_i, g_j)$, причем g_0 – это основная цель команды, а цели $(g_1, \ldots, g_{k-1})$ являются подцелями соответствующего уровня иерархии, а также действиями вышестоящего уровня.

Для параллельного обучения стратегий $\pi_{\theta_{i,j}}$ рассмотрим расширенную иерархическую модель HDec-POMDP как набор из k моделей $(N_j, S_j, \mathbb{A}_j, \mathcal{G}_j, T_j, R_j, \mathbb{O}_j, \Omega_j, h_j, b_j^0)$, где $0 \le j \le k-1$. Каждый уровень иерархии можно представить как отдельную задачу мультиагентного обучения со своими наблюдениями, действиями и вознаграждениями, в то время как каждый агент представляет собой иерархию из k уровней стратегий с передачей подцелей нижестоящим уровням и примитивными действиями на нижнем уровне.

В качестве основы для мультиагентного обучения с подкреплением в данной роботе выбран современный метод MADDPG, который расширил метод DDPG на мультиагентный случай, применил CTDE и продемонстрировал одни из лучших результатов нейросетевого обучения в SMAC. С точки же зрения иерархического обучения, предлагаемый метод является иерархическим исполнителем-критиком и использует воспроизведение ретроспективного опыта для независимого

обучения разных уровней иерархии, снимая ограничения на общее количество уровней. Таким образом, для агента i и уровня иерархии j, где $1 \le i \le n$, $0 \le j \le k-1$, детерминированный градиент стратегии $\mu_{\theta_{i,j}}$, обозначенный как $\mu_{i,j}$, может быть вычислен по формуле:

$$\nabla_{\theta_{i,j}} J(\theta_{\mu_{i,j}}) = \mathbb{E}_{\mu_{i,j}}[\nabla_{\theta_{i,j}} \mu_{i,j}(a_{i,j} \mid o_{i,j}) Q^{\mu}_{i,j}(o_j, a_j, g_j)\big|_{a_{i,j}=\mu_{i,j}(o_{i,j}), g_j=\mu_{i,j-1}(o_{i,j-1})}]. \quad (1)$$

В данной формуле $Q^{\mu}_{i,j}(o_j, a_j, g_j)$ – это совместная функция ценности, зависящая от выполненных действий всех агентов $a_j = (a_{1,j}, \ldots, a_{n,j})$, всех локальных наблюдений $o_j = (o_{1,j}, \ldots, o_{n,j})$ полученных каждым агентом отдельно, а промежуточная цель g_j может заменяться на подцель из буфера воспроизведения ретроспективного опыта на стадии обучения.

4. МЕТОД MASHA

Метод MASHA объединяет в себе способность к временной абстракции, возможность усиления ее эффекта за счет масштабирования добавлением уровней иерархии и способность к децентрализованному исполнению действий множества агентов. Ключевым отличием данного подхода от многих других методов HRL является то, что вместо выбора функции вознаграждения выбирается цель в пространстве состояний.

Достижение этой цели, т.е. приближение к ней в пространстве той же размерности, что и у состояния среды, приносит команде агентов вознаграждение $r = \{0, -1\}$. Это позволяет задать конкретное состояние, в которое нужно привести мультиагентную систему. Минусом можно назвать сложность выбора цели в средах с относительно большим пространством состояний. Поэтому для проведения экспериментов цель выбиралась с использованием экспертных знаний о рассматриваемой среде.

Возврат на более высокий уровень иерархии после выполнения действий уровнем ниже происходит, если подцель достигнута или если закончилось число шагов, отведенных на достижение текущей подцели. Это называется горизонтом планирования каждого уровня и является гиперпараметром H. Само число уровней k – также гиперпараметр.

Одной из самых важных частей подхода является использование ретроспективного опыта согласно принципу HER. Такой механизм коррекции и преумножения опыта взаимодействий со средой позволяет сильно увеличить количество релевантного и полезного для обучения набора траекторий, а также убирает зависимость и проблему нестационарности между разными уровнями иерархии, позволяя тем самым еще и увеличить число возможных уровней. В конце горизонта планирования каждого уровня или при достижении подцели во вспомогательный буфер воспроизведения ретроспективного опыта добавляется копия пройденной агентами траектории с заменой недостигнутой цели на текущее конечное состояние. Это превращает бесполезное и ни к чему не приводящее на первый взгляд взаимодействие со средой в данные, полезные для параллельного обучения всех уровней. В подходе используется модификация HER, позволяющая сохранять для переиспользования произвольное количество цепочек взаимодействий произвольной длины, с одним лишь ограничением длины по общему количеству доступных элементов.

По аналогии с методом HAC, в методе MASHA используется механизм тестирования подцелей, что помогает всем критикам верхних уровней лучше оценивать Q-значения дальних или недостижимых состояний. Тестирование подцелей запускается с некоторой вероятностью λ, что является гиперпараметром. Если во время тестирования агенты не достигают цели на определенном уровне, в буфер воспроизведения попадает транзакция с относительно большим штрафом. Само по себе это не влияет на добавление в буфер воспроизведения также и ретроспективного опыта.

Поскольку каждый агент на каждом уровне, кроме нижнего, выдает подцель для всей команды в качестве своего действия, всему уровню в целом нужно выбрать, какую из всех предлагаемых подцелей выбрать для нижестоящего уровня. Если на нижестоящем уровне агенты получат противоречивые цели, процесс обучения будет невозможен. Для этого вводится дополнительный штраф в виде среднего квадрата отклонений за несоответствие действий верхних уровней, то есть следующих подцелей. Кроме того, в процессе обучения на нижестоящий уровень передается подцель, произведенная лишь одним из агентов, и выбирается он равновероятно из числа всех других.

Множество стратегий одного уровня рассматривается как отдельная задача мультиагентного обучения, в которой к вознаграждению за достижения цели $r = \{0, -1\}$ прибавляется штраф за несоответствие целей разных агентов. Каждый уровень иерархии представлен набором исполнителей и централизованных критиков, что реализовано с использованием метода MADDPG.

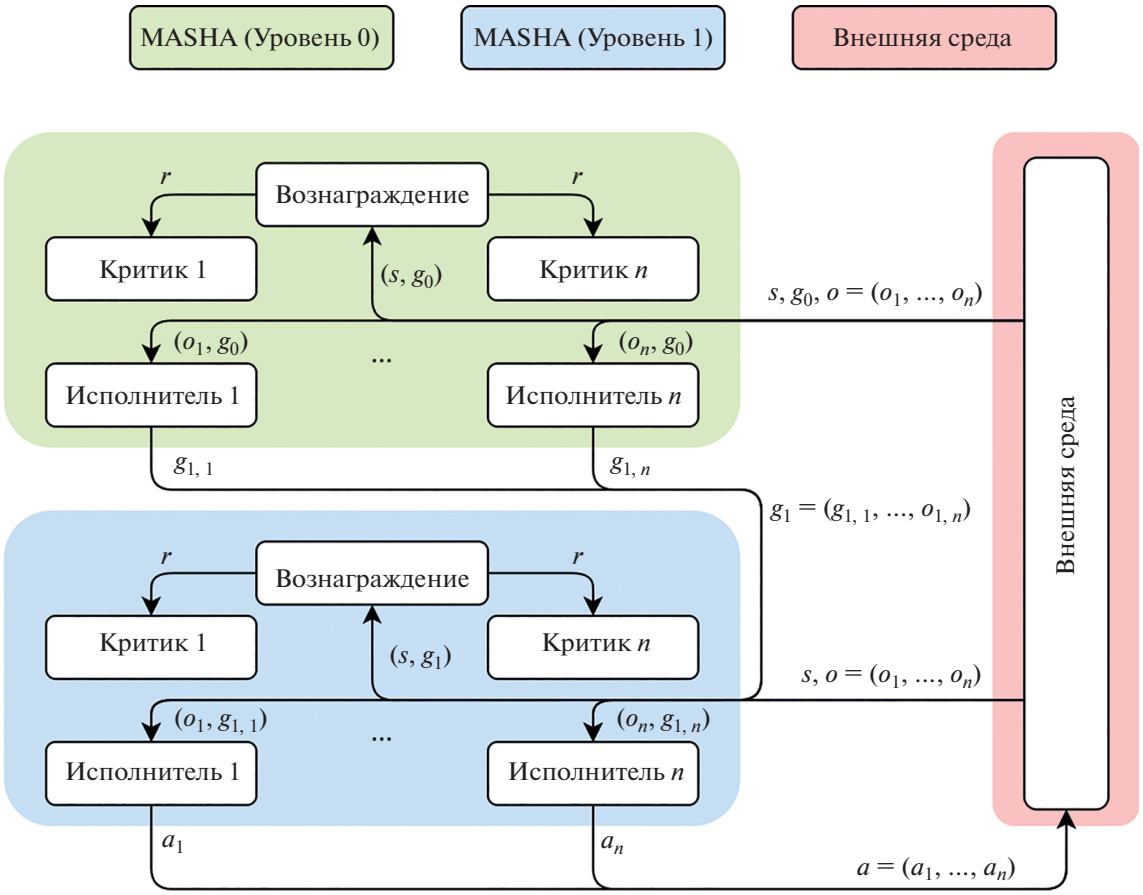

Рис. 1. Архитектура метода MASHA.

На рис. 1 представлена общая архитектура метода MASHA с двухуровневой иерархией. Верхний уровень (Уровень 0) получает начальную командную цель g_0, она одинакова для всех агентов. Исходя из наблюдений агентов и начальной цели, модули исполнителей верхнего уровня генерируют командные подцели $g_1 = (g_{1,1},\ldots,g_{1,n})$ для нижестоящего уровня в качестве своих действий. В процессе обучения для передачи на следующий уровень равновероятно выбирается лишь одна из подцелей, а в процессе тестирования или реального исполнения каждый агент передает и получает между уровнями собственные подцели. Модуль назначения вознаграждения определяет, достигнута ли цель, а также вычисляет меру несоответствия между сгенерированными подцелями, добавляя это к общему вознаграждению. Нижний уровень (Уровень 1) генерирует примитивные действия агентов, которые влияют на внешнюю среду. Модуль вознаграждения этого уровня вычисляет награду лишь на основании факта достижения цели.

Детали обучения иерархии из двух уровней представлены ниже в Алгоритме 1, в котором k – количество уровней, H – горизонт планирования на каждом уровне, λ – вероятность тестирования подцели, γ – дисконтирующий множитель, α_{ac} и α_{cr} – скорости обучения нейросетей исполнителя и критика соответственно, Ne – количество эпизодов обучения, Bs – размер мини-выборки из буферов воспроизведения $\mathcal{D}_0$ и $\mathcal{D}_1$ каждого уровня, вспомогательные буферы ретроспективного опыта $\mathcal{D}_0^{HER}$ и $\mathcal{D}_1^{HER}$ каждого уровня, случайными значениями ξ инициализируются веса основных и целевых нейронных сетей θ и θ', а также шумы $\mathcal{N}_g$ и $\mathcal{N}_a$ для исследования подцелей и примитивных действий соответственно. Указанные численные значения для инициализации алгоритма подобраны экспериментальным путем. Алгоритм использует схему обучения CTDE, относится как к классу алгоритмов вычисления стратегий, так и к классу алгоритмов вычисления ценности состояний-действий, наследует от MADDPG использование случайного шума $\mathcal{N}$ вместо ε – жадного выбора действий и технику мягкой замены весов.

Алгоритм 1. MASHA (2 уровня)

1. **Инициализировать:**

$k \leftarrow 2$, $H \leftarrow 20$, $\lambda \leftarrow 0.3$, $\gamma \leftarrow 0.99$, $\alpha_{ac} \leftarrow 0.001$, $\alpha_{cr} \leftarrow 0.001$,

$Ne \leftarrow 1000000$, $\mathcal{D}_0 \leftarrow 0$, $\mathcal{D}_1 \leftarrow 0$, $\mathcal{D}_0^{HER} \leftarrow 0$, $\mathcal{D}_1^{HER} \leftarrow 0$, $Bs \leftarrow 1024$, $\tau \leftarrow 0.05$,

$\mathcal{N}_g \leftarrow \xi$, $\mathcal{N}_a \leftarrow \xi$, $\theta \leftarrow \xi$, $\theta' \leftarrow \xi$.

2. **Цикл** по шагам от $t_{global} = 1$ до Ne эпизодов обучения:

3. Выбрать командную цель $g_0 \leftarrow \mathcal{G}$.

4. **Цикл** по шагам от $t_0 = 1$ до H или до момента достижения g_0:

5. Для каждого агента $i = (1,\ldots,n)$ получить наблюдение, образуя совместное наблюдение $o_0 = (o_{1,0},\ldots,o_{n,0})$.

6. Для агента i выбрать действие $a_{i,0} = \mu_{\theta_{i,0}}(o_{i,0}, g_0) + \mathcal{N}_g$ с учетом текущей стратегии уровня $\mu_{\theta_{i,0}}$, цели g_0 и случайного шума $\mathcal{N}_g$.

7. Равновероятно выбрать $x \in \{1,\ldots,n\}$ и следующую подцель $g_1 = a_{x,0}$, а также вычислить штраф за несоответствие действий r^{discr}.

8. **Цикл** по шагам от $t_1 = 1$ до H или до момента достижения g_1:

9. Получить совместное наблюдение $o_1 = (o_{1,1},\ldots,o_{n,1})$

10. Определить, будет ли проводиться тестирование подцели с вероятностью λ.

11. Для агента i выбрать действие $a_{i,1} = \mu_{\theta_{i,1}}(o_{i,1}, g_1) + \mathcal{N}$ с учетом текущей стратегии уровня $\mu_{\theta_{i,1}}$, подцели g_1, где случайный шум $\mathcal{N} = 0$, если подцель тестируется, иначе $\mathcal{N} = \mathcal{N}_a$.

12. Выполнить действия $a_1 = (a_{1,1},\ldots,a_{n,1})$, получить новое совместное наблюдение $o_1' = (o_{1,1}',\ldots,o_{n,1}')$ и состояние среды s, проверить, достигнута ли подцель g_1, определить соответствующее вознаграждение $r_1 = \{0,-1\}$ и дисконтирующий множитель $\gamma_1 = \{\gamma, 0\}$.

13. Сохранить $(o_1, a_1, r_1, o_1', g_1, \gamma_1)$ в буфер воспроизведения $\mathcal{D}_1$.

14. Сохранить $(o_1, a_1, None, o_1', None, None)$ в буфер ретроспективного опыта $\mathcal{D}_1^{HER}$, где значения *None* позже будут заменены по принципу HER.

15. **Конец цикла.**

16. Выполнить процедуру HER для буфера $\mathcal{D}_1^{HER}$, обновив $\mathcal{D}_1$.

17. Получить состояние среды s, проверить, достигнута ли цель g_0, определить соответствующее вознаграждение $r_0 = \{0,-1\}$ и дисконтирующий множитель $\gamma_0 = \{\gamma, 0\}$, к r_0 прибавить r^{discr}.

18. При неудачном тестировании подцели сохранить $(o_0, a_0, -H, o_0', g_0, 0)$ в буфер воспроизведения $\mathcal{D}_0$.

19. Сохранить $(o_0, a_0, r_0, o_0', g_0, \gamma_0)$ в буфер воспроизведения $\mathcal{D}_0$.

20. Сохранить $(o_0, a_0, None, o_0', None, None)$ в буфер ретроспективного опыта $\mathcal{D}_0^{HER}$, где значения *None* позже будут заменены по принципу HER.

21. **Конец цикла.**

22. Выполнить процедуру HER для буфера $\mathcal{D}_0^{HER}$, обновив $\mathcal{D}_0$.

23. **Цикл** по уровням от $m = 0$ до $k - 1$:

24. **Цикл** по агентам от $i = 1$ до n:

25. Выбрать случайную выборку (o^j, a^j, r^j, o'^j) размером Bs из буфера $\mathcal{D}_m$, где $j = \{1,\ldots,Bs\}$.

26. Используя целевую нейронную сеть критика уровня m, вычислить:

$$y^j = r_i^j + \gamma Q_i^{\mu'}(o'^j, a_1', \ldots, a_n')\Big|_{a_i' = \mu_i'(o_i^j)}.$$

Рис. 2. Внешняя среда: общий вид малой карты (слева), сценарий 2 против 10 (по центру), использование убежища для достижения победы (справа).

27. Обновить основную нейронную сеть критика уровня m со скоростью обучения α_{cr}, минимизируя функцию потерь:

$$\mathcal{L}(\theta_i) = \frac{1}{Bs}\sum_j (y^i - Q_i^{\mu}(o^j, a_1^j, \ldots, a_n^j))^2.$$

28. Обновить основную нейронную сеть исполнителя уровня m со скоростью обучения α_{ac}, используя градиент:

$$\nabla_{\theta_i} J \approx \frac{1}{Bs}\sum_j \nabla_{\theta_i}\mu_i(o_i^j)\nabla_{a_i} Q_i^{\mu}(o^j, a_1^j, \ldots, a_i, \ldots a_n^j)\Big|_{a_i=\mu_i(o_i^j)}.$$

29. **Конец цикла.**

30. Обновить параметры целевых нейросетей уровня m с помощью техники мягкой замены, используя коэффициент τ:

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'.$$

31. **Конец цикла.**

32. **Конец цикла.**

5. ЭКСПЕРИМЕНТЫ

Для проведения сравнительных экспериментов по мультиагентному обучению с подкреплением была выбрана популярная программная библиотека SMAC, предоставляющая возможность децентрализованного управления множеством агентов в среде стратегической компьютерной игры StarCraft II. Библиотека SMAC сегодня является одним из главных международных экспериментальных стендов для объективного анализа мультиагентных методов машинного обучения.

Каждая союзная управляемая сущность контролируется независимым обучающимся агентом, имеющим доступ лишь к локальным наблюдениям среды. Управляемые сущности противодействующей команды запрограммированы на определенное однообразное поведение. Для наблюдения агентам доступны следующие показатели: очки здоровья, взаимное расположение и тип управляемой сущности в зоне видимости. Вознаграждение назначается агентам только при уничтожении всех сущностей противодействующей команды. Таким образом, среда является мультиагентной и частично наблюдаемой, а вознаграждения редкими.

Для проведения экспериментов была разработана среда, включающая специальный элемент взаимодействия (рис. 2). При одновременном приближении всех союзных агентов к воротам, они открываются, и после попадания агентов внутрь безопасной зоны, ворота закрываются вновь, давая возможность спрятаться от сущностей противодействующей команды, и впредь быть неуязвимыми. Это сильно повышает шансы на победу, однако требует последовательных кооперативных действий в условиях редких вознаграждений. Союзные агенты управляют дальнобойными сущностями типа “морпех” (англ. marine),

которые в начале каждого эпизода появляются в центре карты, а в противодействующей команде сущности ближнего боя типа “зерглинг” (англ. zergling), начинающие эпизод в противоположном от убежища углу.

Реализовано два варианта среды – с картой малого и большого размера, соответственно с меньшим и большим пространством состояний. Обе карты квадратные, но у большей карты утроен размер каждой стороны. В тех экспериментах, где количество союзных сущностей превышало или было равным количеству противодействующих сущностей, оптимальной стратегией оказывалось прямое сражение, в других же, где противодействующих сущностей было больше, оптимальной стратегией являлось совместное попадание агентов в укрытие, где им ничего не угрожало. Поэтому в качестве показательных экспериментов рассмотрим четыре экстремальных сценария (табл. 2).

Для сравнительных экспериментов выбраны следующие методы: MADDPG, LIIR, Independent HAC, HSD. Авторы метода LIIR уделяют внимание проблеме присвоения индивидуального вознаграждения при условии получения общего командного вознаграждения и используют нейросетевую архитектуру исполнитель-критик. Функция внутреннего вознаграждения создается для каждого из агентов, которая по-разному их стимулирует на каждом временном шаге. Причем внутренне вознаграждение отдельного агента используется для обучения собственного вспомогательного критика при формировании индивидуальной политики. Сама же функция внутреннего вознаграждения параметризована и обновляется так, чтобы максимизировать ожидаемое накопленное командное вознаграждение. Independent HAC – это метод обучения, в котором каждый агент в мультиагентной системе использует оригинальную реализацию HAC с двумя уровнями в иерархии, однако кооперация никак дополнительно не настроена.

Методы Independent HAC и MASHA, работающие с целевыми состояниями, получали одну цель для обучения – значения очков здоровья противников в векторе состояния среды должны быть равны нулю, что эквивалентно самому критерию победы в данной среде. Остальные методы использовали при обучении внешний сигнал вознаграждения от среды – 0 или +1, при поражении или победе соответственно.

Во всех экспериментах в качестве оптимизатора параметров нейронной сети использовался оптимизатор Adam [26], обновление параметров целевых нейронных сетей осуществлялось по принципу мягкой замены. Все нейронные сети состояли из полносвязных слоев. Нейронные сети исполнителей состояли из трех слоев с функцией активации ReLU с 64 нейронами на каждом слое, выходной слой с функцией активации Softmax. Нейронные сети критиков состояли из трех слоев с функцией активации ReLU с 128 нейронами на каждом слое, выходной слой с функцией активации Sigmoid. Для реализации нейронных сетей использовался фреймворк pytorch. В экспериментах для методов MADDPG и LIIR использовались гиперпараметры, рекомендуемые их авторами. Программный код разработанных мультиагентных методов с внутренней мотивацией доступен по ссылке: https://github.com/bolshakoVofficial/masha.

Таблица 2. Основные сценарии проведения экспериментов

Количество союзных агентов	Количество агентов-противников	Размер карты
2	2	малая
2	2	большая
2	10	малая
2	10	большая

6. ОБСУЖДЕНИЕ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

В ходе экспериментов оценивалось среднее вознаграждение, получаемое агентами в конце эпизода. Обучение длилось 250 тысяч эпизодов, и для каждого метода проводилось 5 таких запусков с различными случайными начальным значениями (англ. random seed). В целях сравнения для каждого сценария указан оптимальный базовый результат (англ. optimal baseline), принимающий значение максимального возможного вознаграждения (рис. 3).

В сценариях, где количество союзных агентов и противников одинаковое, все методы показали результаты, близкие к оптимальному (табл. 3). Все агенты выучивали стратегию атаки противников. Однако при перевесе по количеству управляемых сущностей в пользу противодействующей команды, только у методов HSD и MASHA получается выучить оптимальную стратегию, занимая убежище на маленькой карте. Это связано с тем, что, используя одноуровневые методы LIIR и MADDPG, агенты испытывают трудности с исследованием среды. Двигаясь случайно или эпсилон-жадно, агенты посещают убежище слишком малое количество раз, или не попадают туда вовсе. Решением может быть использование более продвинутой стратегии исследования среды. Что касается Independent HAC, причина его неудачи вероятнее всего заключается в неспособности координировать действия агентов между собой.

Методы HSD и MASHA по всей видимости не испытывают подобных проблем, поскольку аген-

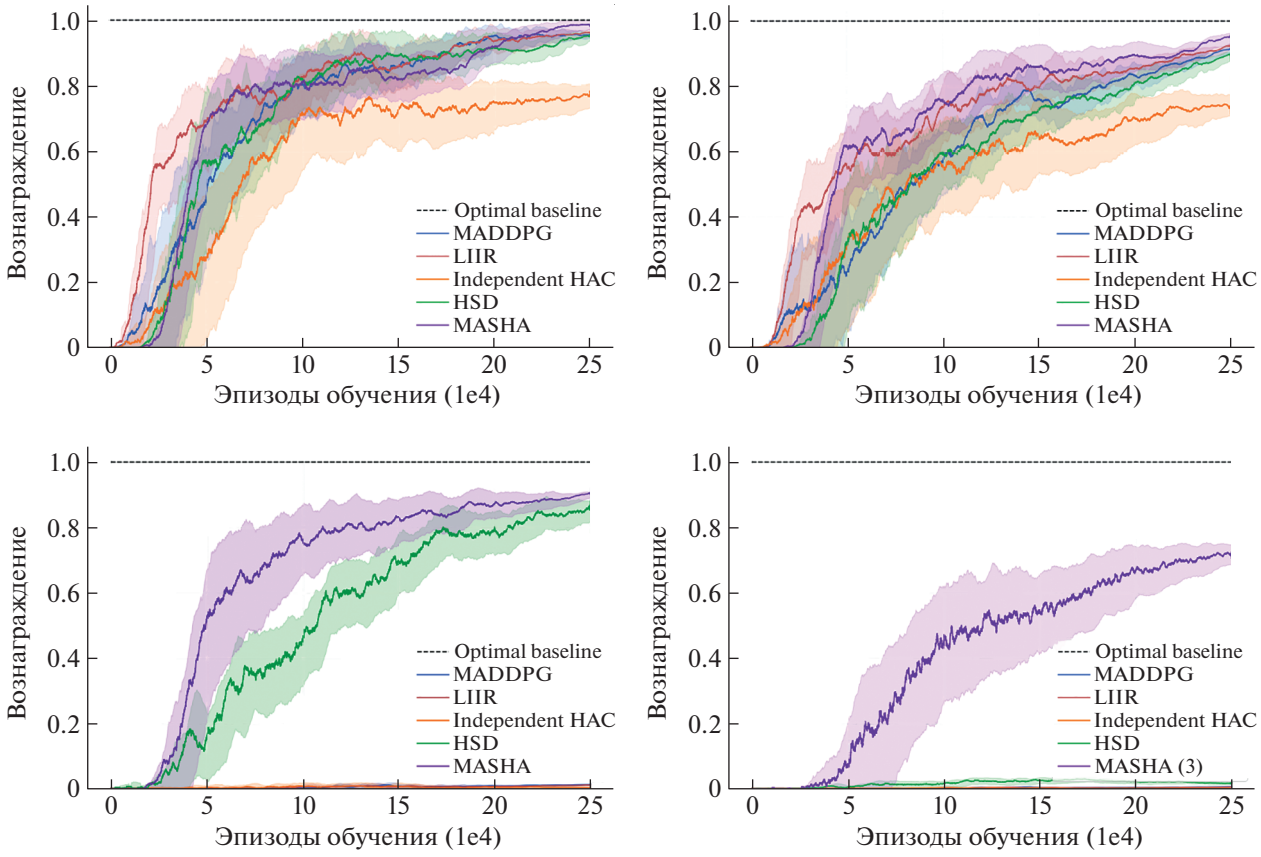

Рис. 3. Результаты обучения в различных сценариях, слева направо и сверху вниз: 2 против 2 на малой карте, 2 против 2 на большой карте, 2 против 10 на малой карте, 2 против 10 на большой карте.

ты способны принимать согласованные решения на разных масштабах времени, и, даже совершая действия с определенной долей случайности, строить продолжительные цепочки действий, приводящие их в редкие состояния. Однако в экспериментах с численным перевесом противников и большой картой только метод MASHA, но уже с тремя уровнями в иерархии способен приблизиться к оптимальному результату. Возможность использовать еще один уровень дополнительно расширяет масштабы времени для принятия решений, а значит делает дальние расстояния в среде более достижимыми, что в связке с координацией агентов позволяет найти оптимальную стратегию в среде с относительно большим пространством состояний.

Таблица 3. Результаты средних вознаграждений в различных сценариях

Метод	Малая карта		Большая карта	
	2 против 2	2 против 10	2 против 2	2 против 10
Optimal baseline	1.0	1.0	1.0	1.0
MADDPG	0.958	0.013	0.919	0.006
LIIR	0.963	0.012	0.926	0.005
Independent HAC	0.784	0.009	0.746	0.003
HSD	0.957	0.879	0.900	0.030
MASHA (2 уровня)	**0.987**	**0.906**	**0.957**	0.040
MASHA (3 уровня)	–	–	–	**0.724**

7. ЗАКЛЮЧЕНИЕ

Сегодня индустриальное применение новых методов мультиагентного машинного обучения – это отдельная и большая задача по портированию программного обеспечения в существующие технологические стеки. К актуальным инновациям индустриального применения методов MAHRL можно отнести автоматический синтез компьютерных программ, повышение эффективности распределения данных на цифровых платформах Интернета вещей, разработку адаптивных пользовательских интерфейсов мобильных устройств, идентификации и решения социальных дилемм в задачах агрегирования веб-сервисов, а также автоматизацию программных средств цифровой обработки изображений. Именно в области компьютерного зрения намечаются главные прорывы по индустриальному внедрению методов MAHRL, так как это среда мультиагентна и иерархична по своей сути, и несомненно имеет нерешенные задачи по разреженной награде, требующие своего внимания со стороны мультиагентного иерархического обучения с подкреплением.

В данной работе проведено исследование алгоритмических процессов объединения иерархического и мультиагентного обучения с подкреплением – двух важнейших парадигм, которые по отдельности продемонстрировали потенциал в решении сложных задач. Предложенный метод MASHA позволяет обучать кооперативных агентов с многоуровневой иерархией. Успешная интеграция модифицированного механизма воспроизведения ретроспективного опыта не только увеличивает эффективность обучения в сложной среде с редкими вознаграждениями, но и улучшает обнаружение промежуточных целей, а также дает возможность параллельного обучения всех уровней иерархии. Эксперименты в цифровой среде StarCraft II продемонстрировали высокую эффективность предлагаемого метода.

Согласованность метода MASHA с парадигмой централизованного обучения с децентрализованным исполнением позволяет добиться координации агентов, оставляя при этом определенную степень автономии, обеспечивая важный для области мультиагентного обучения баланс. Метод MASHA также предлагает относительную свободу в выборе базового алгоритма для мультиагентного обучения, который соответствует парадигме CTDE.

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Работа выполнена при поддержке НИР Госзадание FSFN-2023-0006.

СПИСОК ЛИТЕРАТУРЫ

1. *Singh S., Lewis R., Barto A., Sorg J.* Intrinsically motivated reinforcement learning: An evolutionary perspective // IEEE Transactions on Autonomous Mental Development. 2010. V. 2(2). P. 70–82.
2. *Mnih V., Kavukcuoglu K., Silver D. et al.* Human-level control through deep reinforcement learning // Nature. 2015. V. 518. № 7540. P. 529–533.
3. *Silver D., Huang A., Maddison C. et al.* Mastering the game of Go with deep neural networks and tree search // Nature. 2016. V. 529. P. 484–489.
4. *Vinyals O., Babuschkin I., Czarnecki W.M. et al.* Grandmaster level in StarCraft II using multi-agent reinforcement learning // Nature. 2019. 575. P. 350–354.
5. *Sallab A., Abdou M., Perot E., Yogamani S.* Deep reinforcement learning framework for autonomous driving // Electronic Imaging. 2017. V. 19. P. 70–76.
6. *Yang Y.* Many-Agent Reinforcement Learning // PhD thesis, Department of Computer Science University College London. 2021. 327 p.
7. *Wiering M.* Multi-agent reinforcement learning for traffic light control // In International Conference on Machine Learning (ICML). 2000. P. 1151–1158.
8. *Zheng L. et al.* Episodic multi-agent reinforcement learning with curiosity-driven exploration // Advances in Neural Information Processing Systems. 2021. P. 3757–3769.
9. *Bellemare M., Naddaf Y., Veness J., Bowling M.* The arcade learning environment: An evaluation platform for general agents // In IJCAI, AAAI Press. 2015. P. 4148–4152.
10. *Barto A., Mahadevan S.* Recent advances in hierarchical reinforcement learning // Discr. Event Dyn. Syst. 2003. V. 13. P. 41–77.
11. *Dietterich T.* Hierarchical reinforcement learning with the MAXQ value function decomposition // J. Artif. Int. Res. 2000. V. 13. № 1. P. 227–303.
12. *Sutton R., Precup D., Singh S.* Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning // Artif. Intell. 1999. V. 112. P. 181–211.
13. *Samvelyan M. et al.* The starcraft multi-agent challenge // arXiv preprint arXiv:1902.04043, 2019.
14. *Dayan P., Hinton G.* Feudal reinforcement learning // In Advances in Neural Information Processing Systems. 1993. P. 271–278.
15. *Nachum O., Gu S., Lee H., Levine S.* Data-efficient hierarchical reinforcement learning // In Proceedings of Neural Information Processing Systems. 2018. P. 3307– 3317.
16. *Levy A., Konidaris G., Platt R., Saenko K.* Learning multi-level hierarchies with hindsight // In Proceedings of the 7th International Conference on Learning Representations. 2019. P. 1–15.
17. *Andrychowicz M. et al.* Hindsight experience replay // Advances in neural information processing systems. 2017. P. 1–11.
18. *Bacon P.-L., Harb J., Precup D.* The option-critic architecture // In Proceedings of the 31st AAAI Conference on Artificial Intelligence. 2017. P. 1726–1734.

19. *Yang J., Borovikov I., Zha H.* Hierarchical cooperative multi-agent reinforcement learning with skill discovery // In Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems. 2020. P. 1566–1574.
20. *Rashid T. et al.* QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning // arxiv:1803.11485, 2018.
21. *Ryan L. et al.* Multi-agent actor-critic for mixed cooperative-competitive environments // Advances in neural information processing systems. 2017. P. 1–12.
22. *Yali D. et al.* Liir: Learning individual intrinsic reward in multi-agent reinforcement learning // Advances in Neural Information Processing Systems. 2019. P. 1–12.
23. *Amato C. et al.* Planning for decentralized control of multiple robots under uncertainty // In IEEE International Conference on Robotics and Automation (ICRA). 2015. P. 1241–1248.
24. *Sutton R.S., Barto A.G.* Reinforcement learning: An introduction // MIT press. 2018. 552 p.
25. *Lillicrap T. et al.* Continuous control with deep reinforcement learning // arXiv preprint arXiv:1509.02971, 2015.
26. *Kingma D.P., Adam B.J.* A method for stochastic optimization // arXiv preprint arXiv:1412.6980, 2014.

HIERARCHICAL METHOD FOR COOPERATIVE MULTI-AGENT REINFORCEMENT LEARNING IN MARKOV DECISION PROCESSES

V. E. Bolshakov[a] and A. N. Alfimtsev[a]

[a] *Bauman Moscow State Technical University, Moscow, Russia*

Presented by Academician of the RAS A.A. Shananin

In the rapidly evolving field of reinforcement learning, combination of hierarchical and multi-agent learning methods presents unique challenges and opens up new opportunities. This paper discusses a combination of multi-level hierarchical learning with subgoal discovery and multi-agent reinforcement learning with hindsight experience replay. Combining these approaches leads to the creation of Multi-Agent Subgoal Hierarchy Algorithm (MASHA) that allows multiple agents to learn efficiently in complex environments, including environments with sparse rewards. We demonstrate the results of the proposed approach in one of these environments inside the StarCraft II strategy game, in addition to making comparisons with other existing approaches. The proposed algorithm is developed in the paradigm of centralized learning with decentralized execution, which makes it possible to achieve a balance between coordination and autonomy of agents.

Keywords: Multi-agent reinforcement learning, hierarchical learning, subgoal discovery, hindsight experience replay, centralized learning with decentralized execution, sparse rewards