© 2022 г. А.И. ПАНОВ, канд. физ.-мат. наук (panov.ai@mipt.ru)
(Федеральный исследовательский центр
“Информатика и управление” РАН, Москва;
Московский физико-технический институт
(национальный исследовательский университет))

ОДНОВРЕМЕННОЕ ПЛАНИРОВАНИЕ И ОБУЧЕНИЕ В ИЕРАРХИЧЕСКОЙ СИСТЕМЕ УПРАВЛЕНИЯ КОГНИТИВНЫМ АГЕНТОМ[1]

Задачи планирования поведения и обучения принятию решений в динамической среде в системах управления интеллектуальными агентами обычно разделяют и рассматривают отдельно. Предложена новая объединенная иерархическая постановка задачи одновременно планирования и обучения (SLAP) в контексте предметного обучения с подкреплением и описана архитектура когнитивного агента, решающего данную задачу. Предложен новый алгоритм обучения действиям в частично наблюдаемой внешней среде с использованием подкрепляющего сигнала, предметного описания состояний внешней среды и динамически обновляемых планов действий. Рассмотрены основные свойства и преимущества предложенного алгоритма, среди которых — отсутствие фиксированного когнитивного цикла, вследствие которого ранее приходилось использовать разделение подсистем планирования и обучения, возможность строить и обновлять модель взаимодействия со средой, что повышает эффективность обучения. Предложено теоретическое обоснование некоторых положений данного подхода, предложен модельный пример и продемонстрирован принцип работы SLAP агента при управлении беспилотным автомобилем.

Ключевые слова: обучение с подкреплением, планирование поведения, когнитивный агент, иерархическое планирование, системы управления, беспилотный транспорт, мобильные роботы.

DOI: 10.31857/S0005231022060058, **EDN:** ACLEUU

1. Введение

Современные системы управления беспилотным транспортом и мобильными робототехническими платформами реализуют модульный подход к генерации автономного поведения [1]. Различные подсистемы отвечают за выполнение определенного рода подзадач: генерация траектории движения, реализация предложенной траектории с учетом динамики объекта управления, детекция и сегментирование объектов во внешней среде, планирование действий по манипуляции объектами, обучение модели взаимодействия со средой

[1] Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект № 18-29-22027).

и т.д. При усложнении задач, которые ставятся перед объектом управления, увеличивается количество необходимых подсистем и усложняется их внутренняя организация, усложняется межмодульное взаимодействие.

Однако в последнее время в области разработки общих систем искусственного интеллекта наметилась обратная тенденция по объединению функциональности различных модулей в связи с тем, что для повышения эффективности и адаптивности решения перечисленных выше подзадач требуется комплексирование результатов или во многих случаях одновременная взаимосвязанная работа разных подсистем [2]. Примерами подобных ситуаций могут служить варианты интеграции подсистем компьютерного зрения в задаче управления беспилотным автомобилем, двигающимся в среде с большим количеством других автомобилей и пешеходов, когда для повышения эффективности предсказания траекторий других участников движения необходимо интегрировать в этот модуль работу подсистем сегментации и трекинга объектов [3].

Большое внимание в настоящее время уделяется применению методов машинного обучения в подсистемах, отвечающих как за непосредственное управление движением робототехнической платформы [4], так и за высокоуровневое планирование перемещения и поведения [5]. В данном случае основной задачей является уменьшение роли заранее заданных эвристик и вручную сформированных правил поведения на основе априорных знаний о задаче с целью повышения адаптивности методов и робастности получаемых решений при изменении условий внешней среды.

В настоящей статье предлагается новый подход по интеграции подсистем планирования поведения (т.е. действий как по перемещению, так и, например, манипуляции предметами внешней среды) и обучения поведению, в котором формируется адаптивная стратегия по достижению поставленной перед агентом цели [6]. Такая интеграция является естественной, так как обе подсистемы представляют собой различную реализацию модуля последовательного принятия решений [7]. Однако при планировании необходима модель функционирования внешней среды, а модуль обучения может автоматически формировать такую модель в явном или неявном виде. Для эффективного учета возможностей обеих подсистем предлагается использовать иерархическую организацию как для всей системы управления, так и для разделения высокоуровневого планировщика, для которого уже не требуется полной и точной модели, и низкоуровневой стратегии, которая обучается на основе оригинального метода обучения с подкреплением.

В данном исследовании предложена новая версия иерархической гибридной архитектуры STRL управления сложными техническими объектами [8] с целью выделения подсистем, обучающихся в процессе взаимодействия со средой. На стратегическом уровне управления впервые выделены три подсистемы — предметного представления модели среды, планирования поведения и обучения достижению подцелей. Основным вкладом данной статьи является новый подход к задаче интеграции подсистем планирования и обучения ко-

гнитивного агента, под которым подразумевается мобильная робототехническая платформа или беспилотное транспортное средство. Предлагается оригинальная иерархическая постановка задачи одновременного планирования и обучения (simultaneous planning and learning, SLAP). Во второй части статьи данный подход представлен в виде архитектуры SLAP агента, решающего поставленную задачу на основе иерархического подхода, в котором предлагается использовать планирование поведения по дереву Монте-Карло на верхнем уровне и предметное обучение с подкреплением для частично наблюдаемой среды на нижнем уровне иерархии действий. Представлены теоретическое обоснование для механизма обучения актора и критика в данной постановке. Кроме иллюстративного примера на клеточной среде, предложена схема реализации SLAP агента для задачи управления маневрами беспилотного автомобиля.

2. Архитектура управления поведением STRL2

В условиях динамической среды, свойства и поведение которой заранее не известны когнитивному агенту, в качестве которого будет пониматься мобильная робототехническая платформа, предлагается использовать обновленную версию архитектуры STRL, в которой сделан акцент на возможность обучения как в процессе выполнения действий в среде, так и на предобучение на заранее собранных наборах данных. На рис. 1 представлены схема основных подсистем архитектуры и базовые процедуры управления и передачи информации между модулями. Кратко рассмотрим основные особенности этой архитектуры.

Архитектура STRL2 является иерархической и состоит из трех базовых уровней. На среднем тактическом уровне, также как и в оригинальной версии, решаются задачи классического управления с использованием конкретной модели динамики объекта управления: задачи стабилизации, следования по траектории и т.п. Реактивный уровень осуществляет интегрирование уравнений модели динамики с учетом геометрических ограничений, которые накладываются при планировании траектории на верхнем уровне [9]. Результатом является выработка управляющего сигнала на органы управления.

На тактическом уровне в STRL2 предлагается уделить больше внимания задачам компьютерного зрения, а не только построению и прогнозированию траектории движения во внешней среде. На рис. 1 заполненными блоками отмечены подсистемы, требующие либо интерактивного обучения или предварительного обучения на заранее подготовленных наборах данных или в симуляторах. На тактическом уровне такими обучающимися подсистемами являются подсистемы нейросетевого картирования и локализации (SLAM), сегментации и трекинга объектов во внешней среде по RGB-D изображению или по данным с лазерных дальномеров (лидаров). Данные подсистемы формируют так называемую сенсорную ситуацию, по которой уже возможно построение специфических представлений (графов регулярной структу-

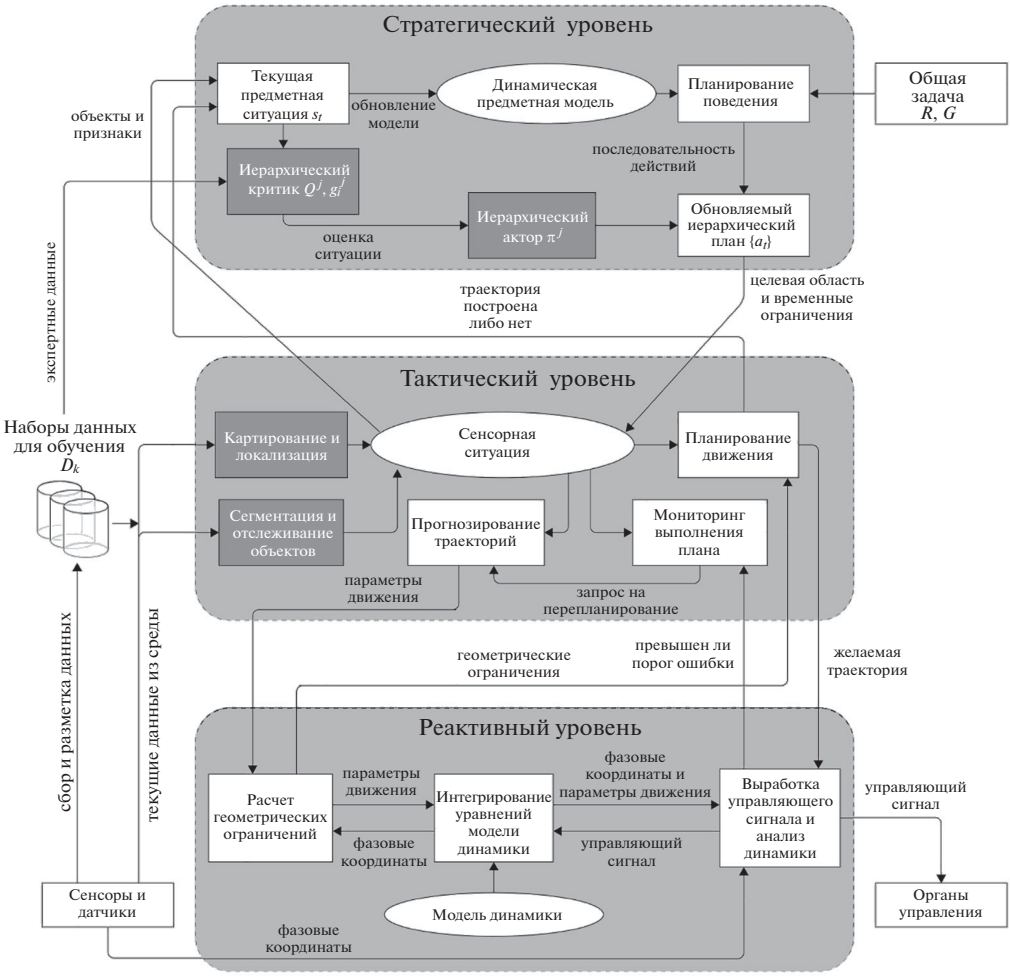

Рис. 1. Основные компоненты и межмодульное взаимодействие в архитектуре STRL2.

ры), необходимых для построения пути и мониторинга движения по траектории [10].

На стратегическом уровне STRL2 информация о текущей сенсорной ситуации используется для обновления долговременной предметной модели, по которой агент строит высокоуровневый план поведения. На этом уровне обучающимися подсистемами являются подсистема формирования оценки текущей ситуации относительно итоговой цели, поставленной перед агентом (модуль критика), и подсистемами актора, который автоматически формирует стратегию по достижению подцелей, выставленных планировщиком по модели. Зачастую критик и актор требуют предобучения в симуляционных средах с использованием заранее заданного сигнала вознаграждения, прежде чем они могут быть использованы для генерации поведения в реальной среде.

В предложенной архитектуре STRL2 делается акцент на формирование адаптивного поведения когнитивного агента в заранее неизвестной динамической среде без необходимости координировать свои действия с другими участниками общей деятельности, в то время как в первой версии архитектуры большее внимание уделялось именно многоагентной составляющей и распределению ролей в коалиции [11].

В подразделе 2.1 будет дана полная постановка задачи работы агента на стратегическом уровне, на котором предлагается объединить в единый цикл работу подсистем планирования и обучения.

Опишем формальную постановку задачи одновременного планирования и обучения с использованием предметно-ориентированной концепции иерархического обучения с подкреплением. Вначале напомним понятие частично наблюдаемого марковского процесса [12], формально представляющего процесс взаимодействия агента и среды, затем расширим его на иерархический случай, введем понятие предметной ситуации и, наконец, объединим это с формальным определением плана действий агента.

2.1. Частично наблюдаемый марковский процесс принятия решений

Итак, пусть $< S, O, A, T, R, G, \Omega >$ — частично наблюдаемый марковский процесс принятия решений (POMDP), где:

- $S = \{s_1, \ldots, s_n\}$ — конечное множество состояний внешней среды, в которой действует агент,
- $O = \{o_1, \ldots, o_k\}$ — конечное множество наблюдений агента, включающих в себя описание объектов (предметов), выделяемых из состояний среды (предполагается, что наблюдение содержит лишь некоторую часть информации о состоянии, т.е. $k < n$),
- $A = \{a_1, \ldots, a_m\}$ — конечное множество действий, в том числе и составных,
- $T : S \times A \to \Pi(S)$ — функция переходов, определяющая по текущему состоянию и действию распределение вероятностей на состояниях среды в следующий момент времени (здесь и далее будем обозначать через $\Pi(X)$ множество вероятностных распределений на конечном множестве X),
- $R : S \times A \to \mathbb{R}$ — функция вознаграждений,
- $G : S \to \{0, 1\}$ — целевая функция, определяющая момент остановки эпизода взаимодействия,
- $\Omega : S \times A \to \Pi(O)$ — функция наблюдений, определяющая распределение вероятностей для наблюдений в текущем состоянии.

В общей постановке задачи предполагается, что все функции в определении частично наблюдаемого марковского процесса принятия решений (POMDP) T, R, G, Ω агенту неизвестны и он может лишь оценивать их в результате взаимодействия со средой, выполняя некоторое действие $a_t \in A$ и получая из среды некоторое наблюдение $o_{t+1} \in O$ и вознаграждение $r_{t+1} = R(s_t, a_t)$. Таким образом, агенту заранее известно только множество A, множество O он может восстановить в явном виде в процессе взаимодействия со средой, а множество S он может оценить только по наблюдениям.

Целью агента является построение такой функции $\pi : O \to \Pi(A)$, задающей вероятностное распределение на множестве действий A при условии текущего наблюдения $o \in O$, при котором максимизируется ожидаемое суммарное вознаграждение (отдача):

$$\mathbb{E}_\pi \left[\sum_{t:G(s_t)\neq 1} \gamma^t R(s_t, a_t) \right] \to \max_\pi, \tag{1}$$

где γ – дисконтирующий множитель. Здесь предполагается, что суммирование идет до тех пор, пока целевая функция $G(s_t)$ не примет значение единица. Подсчет математического ожидания по стратегии подразумевает усреднение по траекториям в пространстве состояний, по которым считается отдача. Ожидаемую отдачу можно подсчитать и для каждого состояния (функция полезности $V(s)$), и для пары состояние-действие (функция $Q(s, a)$).

Функция $\pi(o|s)$ называется стратегией агента и служит для определения последовательности действий агента по заданной последовательности состояний среды. В постановке задачи безмодельного обучения с подкреплением в частично наблюдаемой среде предполагается реактивное поведение агента, при котором агент не прогнозирует реакцию среды в каждый момент времени t и генерирует новое действие $\mathbb{E}_\pi \sum_{t=0}^{\tau} \gamma^t R(s_t, a_t)$ в предположении, что вся существенная информация для принятия решения содержится в состоянии $\mathbb{E}_\pi \sum_{t=0}^{\tau} \gamma^t R(s_t, a_t)$, которое он определяет на основе текущего наблюдения o_t. Вероятность пребывания в следующем состоянии среды $b_{t+1}(s_{t+1})$ (предполагаемое состояние) определяется агентом по текущему наблюдению o_t в соответствии с выражением:

$$b_{t+1}(s_{t+1}|o_{t+1}) = \frac{\Omega(o_{t+1}|s_{t+1}, a_t) \sum_{s_t} T(s_{t+1}|s_t, a_t) b_t(s_t)}{\sum_{s_{t+1}} \left(\Omega(o_{t+1}|s_{t+1}, a_t) \sum_{s_t} T(s_{t+1}|s_t, a_t) b_t(s_t) \right)}. \tag{2}$$

Используя понятие предполагаемого состояния b_t, можно ввести обычный марковский процесс принятия решений на непрерывном множестве таких состояний. В этом случае функции полезности будут определяться для предполагаемых состояний b_t:

$$V^\pi(b_t) = \mathbb{E}_\pi \sum_t \gamma^t \sum_s b_t(s) R(s, a_t). \tag{3}$$

2.2. Иерархическая постановка и параметризация

Введем иерархию на множестве действий, следуя концепции полумарковского процесса принятия решений и подхода умений [13]. Введем так называемое длящееся во времени действие, или умение, $\kappa =< I_\kappa, \pi_\kappa, \beta_\kappa >$, где $I_\kappa \subseteq O$ – инициирующее множество наблюдений, π_κ – стратегия, реализующая данное

умение, $\beta_\kappa : O \to \{0,1\}$ – терминальная функция, останавливающая реализацию умения. Множество умений будем обозначать через κ. Расширение множества действий за счет множества умений приводит к определению полумарковского процесса принятия решений и введению функций полезности состояния $V_\kappa(b)$ и умения $Q_\kappa(b,\kappa)$.

В иерархической постановке агент должен сформировать как стратегию π_κ на множестве умений (высокоуровневая стратегия) внутренние стратегии для каждого умения π_κ (низкоуровневые стратегии) и функции остановки для каждого умения β_κ. Не снижая общности постановки задачи, можно считать, что инициирующие множества для всех умений включают все возможные наблюдения $I_\kappa = O$. Будем параметризовать стратегию π_κ и функцию остановки β_κ с помощью наборов параметров θ и ϑ соответственно. В иерархической постановке цель агента – максимизировать отдачу, начиная с предполагаемого состояния b_0 и умения κ_0, — запишется в виде

$$\mathbb{E}_{\kappa,\theta,\kappa}\left[\sum_{t:G(s_t)\neq 1}\gamma^t\sum_s b_t(s)R(s,a_t)\,\middle|\,b_0,\kappa_0\right]\to\max_{\theta,\vartheta}. \tag{4}$$

Определим полезность выполнения конкретного действия a в рамках умения κ в предполагаемом состоянии $b(s)$ (Q_a-функция) и полезность самого умения κ в $b(s)$ (Q_κ-функция). Полезность умения определяется его внутренней стратегией и полезностью каждого действия: $Q_\kappa(b,\kappa) = \sum_a \pi_{\kappa,\theta}(a|o)Q_a(b,\kappa,a)$, где полезность действия в свою очередь записывается через функцию переходов среды (здесь и далее через штрих будем обозначать следующий момент времени):

$$\begin{aligned} Q_a(b,\kappa,a) &= \sum_s b(s|o)R(s,a) + \\ &+\gamma\sum_{o'}\left(\sum_{s'}\Omega(o'|s',a)\sum_s T(s'|s,a)b(s|o)\right)\tilde{Q}_\kappa(b'(s'|o),\kappa). \end{aligned} \tag{5}$$

В определении полезности действия Q_a используется поправка к полезности умения $\tilde{Q}_\kappa$, которая учитывает возможность остановки умения при следующем наблюдении o':

$$\tilde{Q}_\kappa(b',\kappa) = \big(1-\beta_{\kappa,\vartheta}(o')\big)\,Q_\kappa(b',\kappa) + \beta_{\kappa,\vartheta}(o')V_\kappa(b'). \tag{6}$$

2.3. Предметная ситуация

Наблюдение, получаемое агентом, практически во всех значимых окружениях представляет собой некоторую сцену, состоящую из объектов или предметов. Декомпозиция предметной сцены на отдельные взаимосвязанные составляющие может оказаться полезной в том случае, когда такие взаимосвязи отделимы от самих предметов, а действия агента могут быть отнесены

не ко всей сцене, а к концертному целевому предмету. Формально такая декомпозиция для марковского процесса принятия решений описывается в так называемой объектно-ориентированной поставке [14, 15]. В постановке задачи одновременного обучения и планирования будет рассматриваться случай, когда взаимосвязи объектов несущественны для принятия решений агентом и принципиально только наличие тех или иных предметов в сцене.

Итак, пусть и состояние среды s, и наблюдение агента o представляют собой некоторое множество независимых объектов, которые относятся к конечному числу классов $C = \{c_1, c_2, \ldots, c_k\}$. Каждый класс характеризуется своим набором атрибутов или признаков $\{f_1^c, f_2^c, \ldots, f_{n_c}^c\}$, а объект $e \in E$ класса $c(e) \in C$ описывается конкретными значениями данных признаков $e = \{d_1^c, d_1^c, \ldots, d_{n_c}^c\}$. Как состояние s, так и наблюдение агента o, таким образом, представляет собой объединение состояний объектов $\bigcup_{i=1}^{n} e_i$. Будем считать, что частичная наблюдаемость выражается в том, что состояние s и наблюдение o отличаются друг от друга набором объектов и (или) значениями характеризующих их признаков. Выделение объектов по наблюдению происходит с помощью некоторой функции $\Phi^p : O \to 2^E$, которая будет считаться заранее заданной.

В предположении независимого присутствия предметов в среде в текущем наблюдении возможна декомпозиция функции переходов и функции вознаграждений по отдельным классам объектов: $T = \{T_{c_i} | c_i \in C\}$, $R = \{R_{c_i} | c_i \in \in C\}$. Низкоуровневая стратегия агента будет состоять из действий, условиями выполнения для которых будет служить наличие объекта определенного класса, т.е. множество действий также разбивается на подмножества в соответствии с количеством классов $A = \{A_{c_i} | c_i \in C\}$. Проведя такую декомпозицию задачи, возможно определить и выписать соотношение на полезность не всего состояния или наблюдения, а на полезность конкретного объекта, имеющегося в текущем наблюдении. Все соотношения на функции полезности, определенные в подразделе 2.4, остаются в силе с той лишь поправкой, что в текущем наблюдении агент выбирает действие жадно, в соответствии с наибольшей полезностью конкретного объекта $Q_a(b, \kappa, a) = = \arg\max_{e \in o} Q_a(e, \kappa, a_{c(e)})$. Будем считать, что умения агента не поддаются аналогичной декомпозиции и зависят от всего наблюдения целиком.

2.4. Обновление плана поведения

В постановке задачи одновременного обучения и планирования агент автоматически строит обновляемую модель среды $M =< \hat{T}, \hat{R}, \hat{\Omega} >$, которая позволяет находить план B достижения цели $G(s_l) = 1$ за счет моделирования переходов при некоторой модельной стратегии π:

$$B^\pi =< o_0, r_0, a_0, o_1, r_1, a_1, \ldots, a_{l-1}, o_l >,$$

где $s_i = \hat{T}(s_{i-1}, a_{i-1})$, $r_i = \hat{R}(s_i, a_i)$, $a_i \sim \pi(a|o_i)$, а заключительное наблюдение o_l соответствует целевому состоянию s_l согласно функции $\hat{\Omega}$. Здесь

под $\hat{T}, \hat{\Omega}$ и $M =< \hat{T}, \hat{R} >$ будем понимать приближенные (аппроксимируемые) значения функций переходов, наблюдений и вознаграждений соответственно. План поведения агента, таким образом, составляется жадным образом в точности до небольшой поправки, отвечающей за исследование среды, в предположении корректности текущего приближения модели $M =< \hat{T}, \hat{R} >$.

Под процессом обучения агента будет пониматься итерационное обновление модели $M =< \hat{T}, \hat{R} >$, функций полезности Q, стратегии π и соответственно плана поведения B^{π}. Возможны четыре основных варианта составления общей схемы обучения агента с использованием фазы планирования по модели. Приведем краткие алгоритмические схемы для этих вариантов. Будем обозначать собранный агентом опыт в виде множества прецедентов $D = \{(o_t, r_t, a_t)_{t=1}^{T}\}$. Траекторией будем называть некоторую последовательность таких прецедентов в порядке их формирования при взаимодействии со средой. Первый вариант интеграции представляет собой обучение с использованием планируемых ("воображаемых") траекторий [16]:

1. Агент предсказывает ("воображает") траектории с некоторого состояния s_t, используя модель $M = < \hat{T}, \hat{R} >$.
2. Агент обновляет свою стратегию, используя прецеденты из предсказываемых траекторий.
3. Агент набирает новый опыт взаимодействия со средой по обновленной стратегии.
4. По собранному опыту D агент обновляет модель среды $M = < \hat{T}, \hat{R} >$.
5. Шаги 1–4 повторяются до сходимости.

Второй вариант — обучение с использованием разделения стратегий — подразумевает использование модели только на начальной стадии взаимодействия со средой, постепенно расширяя горизонт ее применения в процессе уточнения:

1. Агент планирует и выполняет первые шаги в траектории, используя модель $M = < \hat{T}, \hat{R} >$.
2. Агент использует текущую стратегию для продолжения траекторий в процессе взаимодействия со средой.
3. По собранному опыту агент обновляет модель и стратегию, одновременно выбирая критерий остановки планирования и запуска интерактивной стратегии.

Обучение с имитацией эпизодов возможно только в случае наличия копий среды, в которых агент имитирует свое поведение:

1. Используем возможность запускать симуляции действий агента в среде, чтобы с текущего шага проиграть некоторое количество эпизодов для обновления этой модели.
2. На основе обновленной модели определяем полезность состояний и выбираем действие, выполняемое в среде.
3. Обновляем стратегию, используя выбранное действие в качестве эталонного.

В предлагаемой в данной статье постановке задачи одновременного обучения и планирования используется иерархия для разделения применения модели и интерактивной стратегии:

1. Агент на верхнем уровне иерархии действия использует модель $M = = < \hat{T}, \hat{R} >$ для получения плана на множестве умений.
2. Стратегия каждого умения формируется в интерактивном режиме в процессе взаимодействия со средой.
3. Набранный опыт используется агентом для одновременного уточнения модели на умениях и стратегий каждого умения.

Принимая во внимание все приведенные уточнения постановки задачи одновременного обучения и планирования, получается иерархическая постановка, в которой агенту необходимо максимизировать получаемую в рамках эпизода отдачу с возможностью декомпозиции наблюдения по предметному принципу, автоматическому построению стратегий умений и при одновременном автоматическом формировании модели среды, используемой для планирования на множестве умений.

3. Архитектура SLAP агента

Для решения поставленной в разделе 2 общей задачи одновременного обучения и планирования в данной статье предлагается использовать следующую архитектуру интеллектуального SLAP агента (см. рис. 2). Подсистему обучения агента разделим на две составляющие, как это принято в теории обучения с подкреплением. Критик обновляет функцию полезности действия Q_a, а актор формирует стратегию π_κ в рамках текущего умения. Цикл взаимодействия SLAP агента со средой будет выглядеть следующем образом:

1. Агент использует текущую модель для того, чтобы сформировать план $B^\pi = < o_0, r_0, a_0, o_1, r_1, a_1, \ldots, a_{l-1}, o_l >$ на множестве умений с помощью процедуры SLAPplan. В общем случае предполагается, что уровней иерархии действий может быть несколько (вложенные умения) и может быть сформировано несколько вложенных планов: от высокоуровневого до низкоуровневого. Далее считаем, что план B^π является низкоуровневым.
2. В соответствии с планом агент B^π выбирает текущее умение κ_t.
3. Агент получает из среды текущее наблюдение, которое с помощью функции Φ^p переводится в набор предметов $o_t \to \{e_1, \ldots, e_n\}$.
4. В соответствии со стратегией π_κ для умения κ_t агент выбирает текущее действие a_t.
5. Агент выполняет действие a_t в среде и получает новые наблюдения и вознаграждение.
6. С помощью процедуры SLAPlearn агент проводит оценку выполненного действия и самого умения с помощью критика, а затем обновляет функции аппроксимации критика и актора.
7. Если текущее умение завершилось, выполняется перепланирование, и затем происходит переход к шагу 3.

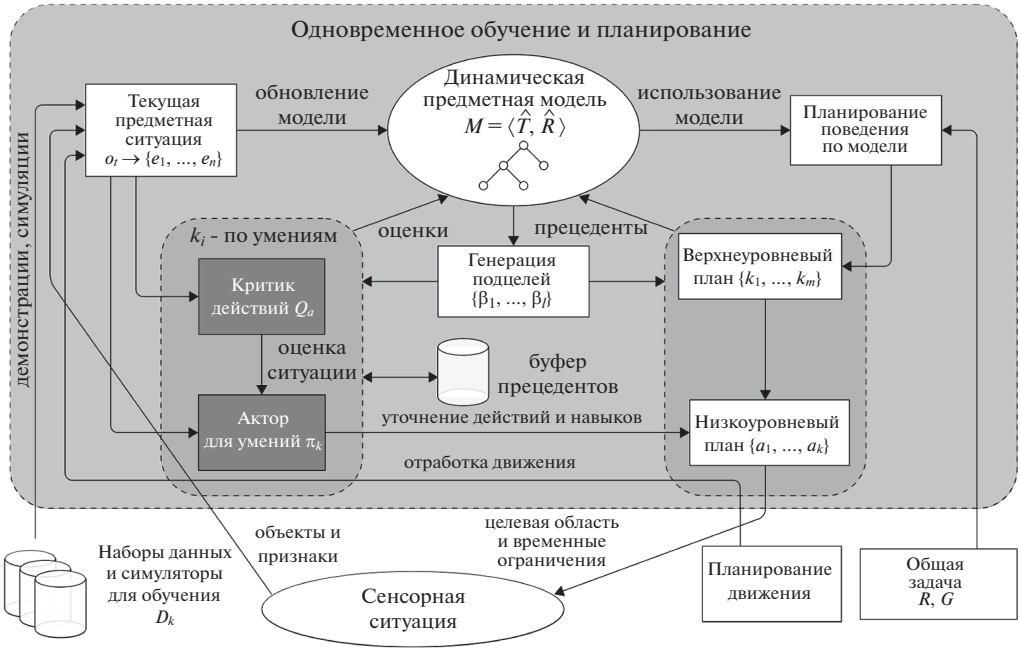

Рис. 2. Архитектура SLAP агента с подсистемами обучения (выделены темным цветом) и планирования поведения. Данная схема представляет собой верхний уровень архитектуры STRL, которая изображена на рис. 1, с детализацией взаимодействия методов обучения и планирования.

Кратко изложим принципы работы процедур SLAPplan и SLAPlearn. При планировании агент использует модель $M = < \hat{T}, \hat{R}, \hat{\Omega} >$ для построения плана своего поведения. В настоящей статье предлагается использовать реализацию модели в виде расширенного дерева поиска Монте-Карло, где каждый узел дерева отвечает за конкретный объект, выделяемый из наблюдения. Ребро дерева соответствует выбору некоторого объекта из наблюдения, выполнению некоторого действия (умения) и переходу к наблюдению, где выделяется следующий объект. В случае, когда для выполнения выбирается действие (умение), для которого в модели неизвестно следующее наблюдение, образуются новые узлы с соответствующими объектами, выделенными из наблюдения, полученного из среды.

Планирование $SLAPplan(M, e)$ в данном дереве $M = < \hat{T}, \hat{R}, \hat{\Omega} >$ происходит за счет поиска кратчайшего пути с учетом дополнительного веса для действий, направленных на исследование среды:

1. Выбираем планируемое умение $\kappa \leftarrow \arg\max_{\kappa} Q_\kappa(e) + \eta\sqrt{\frac{\log N(e)}{N(e,\kappa)}}$. Здесь второе слагаемое отвечает за верхнюю доверительную границу (UTC) эффективной стратегии исследования среды, η – константа, N – счетчики.
2. Производим переход по дереву $e' \leftarrow \hat{T}_c(\kappa)$, $r \leftarrow \hat{R}_c(\kappa)$, где c – класс объекта e.

3. Производим планирование для следующего объекта с подсчетом получаемого вознаграждения $\tilde{R} \leftarrow r + \gamma SLAPplan(M, e')$.
4. Обновляются счетчики $N(e, \kappa) \leftarrow N(e, \kappa) + 1$, $N(e) \leftarrow N(e) + 1$.
5. Настраивается модель — обновляем полезность умения $Q_\kappa(b, \kappa) \leftarrow$ $\leftarrow Q_\kappa(b, \kappa) + \frac{\tilde{R} - Q_\kappa(b,\kappa)}{N(e,\kappa)}$, где b – предполагаемое состояние, для которого выделяется объект e.

В процедуре обучения критика и актора SLAPlearn производится обновление как критерия достижения подцели β_κ, так и стратегии π_κ конкретного, выбранного на верхнем уровне иерархии умения. Напомним, что здесь используется параметризация с помощью набора параметров ϑ для условия завершения и набора θ — для стратегии:

1. Агент выбирает действие в соответствии с текущей стратегией умения $a \sim \pi_{\kappa,\theta}(a|o)$.
2. Агент выполняет действие a, наблюдает o' и r.
3. Критик обновляет оценку полезности действия в рамках текущего умения:

$$Q_a(b, \kappa, a) \leftarrow Q_a(b, \kappa, a) +$$
$$+ \alpha \left(r + \gamma \left((1 - \beta_{\kappa,\vartheta}(o'))Q_\kappa(b', \kappa) + \beta_{\kappa,\vartheta}(o') \max_{\kappa'} Q_\kappa(b', \kappa') \right) - Q_a(b, \kappa, a) \right),$$

   где α – шаг обучения критика, g – обновляемое целевое значение для критика.
4. Актор обновляет параметры для стратегии и для подцели

$$\theta \leftarrow \theta + \alpha_\theta \nabla_\theta \log \pi_{\kappa,\theta}(a|o) Q_a(b, \kappa, a),$$
$$\vartheta \leftarrow \vartheta + \alpha_\vartheta \nabla_\vartheta \tilde{Q}_\kappa(b', \kappa),$$

   где α_θ и α_ϑ – шаги обучения.
5. Обновляем значение градиента полезности умения $\nabla_\theta Q_\kappa$, который может быть использован на верхнем уровне иерархии.
6. Если в соответствии с $\beta_{\kappa,\vartheta}$ подцель достигнута, то в соответствии с высокоуровневым планом выбирается новое умение κ.

Здесь предполагается, что полезность текущего умения передается из подсистемы планирования, где обновление полезности происходит во время обновления самой модели. В разделе 5 дан вывод выражений для градиента $\nabla_\theta Q_\kappa$ (теорема о градиенте критика) и для градиента $\nabla_\vartheta \tilde{Q}_\kappa$ генератора подцелей (теорема о градиенте актора).

4. Модельный пример

В качестве модельного примера рассмотрим задачу обучения навигации до некоторой заданной целевой точки в клеточной среде с препятствиями и управляемыми объектами (дверьми) (см. рис. 3,*a*). Среда организована в виде комнат с узкими проходами между ними таким образом, чтобы поддерживать формирование двухуровневой иерархии действий. Верхний уровень:

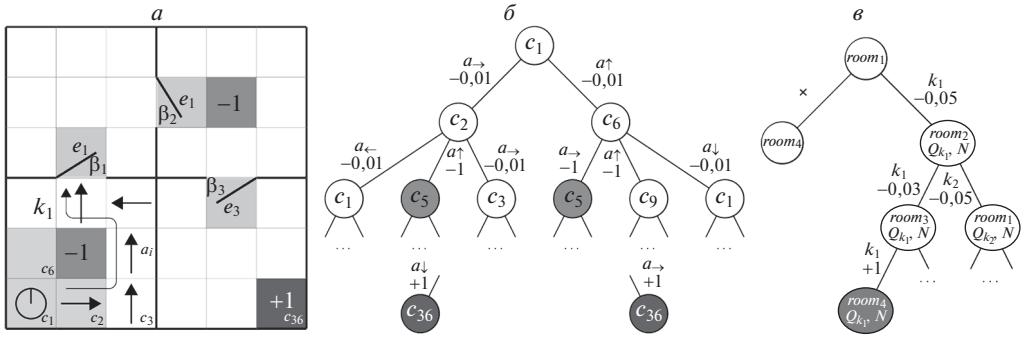

Рис. 3. *а* – Модельный пример перемещения агента по клеточной среде с комнатами, открывающимися проходами между ними и опасными состояниями, в которых завершается эпизод взаимодействия; *б* – соответствующая марковскому процессу принятия решения модель, которая строится агентом без иерархической декомпозиции; *в* – модель клеточной среды, которую строит агент с умениями.

умения по переходу от одной комнаты к другой, нижний уровень: действия по перемещению в четыре стороны для достижения выхода из комнаты или достижения целевой клетки в комнате. Опишем среду подробнее.

В данном примере наблюдение агента o_t — это область видимости вокруг агента радиусом в две клетки (на рис. 3 обозначены серым цветом). Множество действий агента A состоит из четырех действий: поворот на 90° по часовой или против часовой стрелки, проход прямо, открытие двери. На верхнем уровне иерархии агенту доступны умения κ_i, состоящие из действий a_i. Подцелями β_i, в которых завершаются умения, являются клетки, обозначенные светло-серым цветом. Выделяемыми с помощью функции Φ^p объектами e_i в среде являются сами комнаты, двери каждой комнаты, принадлежащие классу объектов c_0, и отдельные клетки, в которых в данный момент находится агент. При этом каждой комнате и клетке соответствует свой класс c_i. Функция вознаграждения R реализована следующим образом: каждый момент времени агенту дает небольшое отрицательное вознаграждение $(-0,01)$, за попадание в темные клетки — (-1) с завершением эпизода, за достижение цели (темная клетка справа внизу) дается $+1$. В качестве выбираемой параметризации используется линейная модель, взвешивающая признаки, соответствующие предметам, выделяемым по наблюдению.

Предметная модель, которую строит и обновляет агент, представлена на рис. 3,*в*. Для сравнения на рис. 3,*б* показана модель, которую агент аппроксимировал бы без иерархического представления действий. Процедуры планирования $SLAPplan(M, e)$ и обучения SLAPlearn для агента для представленного пример будут выглядеть следующим образом.

SLAPplan:

1. Выбираем планируемое умение κ по переходу из текущей комнаты e в соседнюю e' в соответствии с предсказываемой по дереву полезностью данного умения с учетом параметра исследования среды.

2. По текущему виду дерева (так как модель обновляется в процессе обучения, она может не соответствовать итоговой модели на рис. 3,*в*) производится переход в соседнюю комнату e' по стратегии умения π_κ с подсчетом вознаграждения в виде суммы вознаграждений после каждого действия стратегии π_κ.
3. Рекурсивно запускается аналогичная процедура для новой комнаты e'.
4. Обновляются счетчики $N(e,\kappa) \leftarrow N(e,\kappa)+1$, $N(e) \leftarrow N(e)+1$.
5. Настраивается модель — обновляется полезность умения $Q_\kappa(b,\kappa)$.

SLAPLearn:

1. Агент выбирает действие по переходу в соседнюю клетку или открытию двери в соответствии с текущей стратегией умения $a \sim \pi_{\kappa,\theta}(a|o)$, учитывая, что сейчас агент находится в определенной комнате и выполняется текущее умение κ.
2. Агент выполняет действие a, обновляется наблюдение — в область видимости агента попадают новые предметы (клетки, дверь).
3. Критик обновляет оценку полезности действия в рамках текущего умения, вычисляя TD ошибку.
4. Актор обновляет параметры для стратегии и для подцели, вычисляя градиент.
5. Если в соответствии с $\beta_{\kappa,\vartheta}$ подцель достигнута, то в соответствии с высокоуровневым планом выбирается новое умение κ.

5. Теоремы о градиенте

В разделе 2 была представлена общая схема взаимодействия SLAP агента со средой, где для обучения агента необходимо знание целевого значения критика g и для градиента функции полезности стратегии ∇J. Проведем краткие выкладки для вычисления их значений.

Выражение $\sum_{s'} \Omega(o|s',a) \sum_s T(s'|s,a) b(s|o)$ представляет собой вероятность получения агентом следующего наблюдения o', что будем обозначать как $p(o|a,b)$. В начале найдем выражение для градиента полезности умения $Q_\kappa(b,\kappa)$ относительно параметров реализующей его стратегии θ:

$$
\begin{aligned}
(7)\quad \nabla_\theta Q_\kappa(b,\kappa) &= \nabla_\theta \sum_a \pi_{\kappa,\theta}(a|o) Q_a(b,\kappa,a) = \\
&= \sum_a \left(\nabla_\theta \pi_{\kappa,\theta}(a|o)\right) Q_a(b,\kappa,a) + \\
&+ \sum_a \pi_{\kappa,\theta}(a|o) \nabla_\theta \left(\sum_s b(s|o) R(s,a) + \gamma \sum_o p(o|a,b) \tilde{Q}_\kappa(b'(s'|o),\kappa) \right) = \\
&= \sum_a \left(\nabla_\theta \pi_{\kappa,\theta}(a|o)\right) Q_a(b,\kappa,a) + \sum_a \pi_{\kappa,\theta}(a|o) \sum_{o'} \gamma p(o|a,b) \nabla_\theta \tilde{Q}_\kappa(b'(s'|o),\kappa).
\end{aligned}
$$

Используя выражение в определении поправленной полезности $\tilde{Q}_\kappa$, найдем ее градиент:

$$\nabla_\theta \tilde{Q}_\kappa(b',\kappa) = (1-\beta_{\kappa,\vartheta}(o')) + \sum_{\kappa'} \left(\beta_{\kappa,\vartheta}(o')\pi(\kappa'|b')\right) \nabla_\theta Q_\kappa(b',\kappa'). \tag{8}$$

Подставляя это в выражение для градиента полезности умения, получим:

$$\begin{aligned}\nabla_\theta Q_\kappa(b,\kappa) &= \sum_a \left(\nabla_\theta \pi_{\kappa,\theta}(a|o)\right) Q_a(b,\kappa,a) + \\ &+ \sum_a \pi_{\kappa,\theta}(a|o) \sum_{o'} \gamma p(o|a,b) \nabla_\theta \tilde{Q}_\kappa(b'(s'|o),\kappa) = \\ &= \sum_a \left(\nabla_\theta \pi_{\kappa,\theta}(a|o)\right) Q_a(b,\kappa,a) + \sum_{o'}\sum_{\kappa'} p(o',\kappa'|b,\kappa) \nabla_\theta Q_\kappa(b',\kappa'),\end{aligned} \tag{9}$$

где $p(o',\kappa'|b,\kappa)$ задает расширенный марковский процесс принятия решений, в котором состояниям соответствует пара наблюдение-умение. Учитывая марковское свойство при раскрытии рекурсии в определении $\nabla_\theta Q_\kappa$, получается доказательство теоремы 1.

Теорема 1 (о градиенте критика умений). *Для фиксированного множества марковских умений со стохастической реализующей стратегией, дифференцируемой по параметрам θ, градиент ожидаемой дисконтированной отдачи по параметрам θ с начальными условиями (b_0,κ_0) равен*

$$\nabla_\theta Q_\kappa(b,\kappa) = \sum_{b,\kappa} \mu_\kappa(b,\kappa|b_0,\kappa_0) \sum_a \left(\nabla_\theta \pi_{\kappa,\theta}(a|o)\right) Q_a(b,\kappa,a), \tag{10}$$

где $\mu_\kappa(b,\kappa|b_0,\kappa_0)$ – дисконтированные частоты появления предполагаемого состояния и умения по траекториям, начинающимся с начальных условий (b_0,κ_0).

В разделе 4 для обновления параметров генератора подцелей было указано на необходимость вычисления градиента $\nabla_\vartheta \tilde{Q}_\kappa$. Используем определение для этой функции полезности:

$$\begin{aligned}\nabla_\vartheta \tilde{Q}_\kappa &= \nabla_\vartheta \beta_{\kappa,\vartheta}(o')(V_\kappa(b') - Q_\kappa(b',\kappa)) + \\ &+ (1-\beta_{\kappa,\vartheta}(o')) \sum_a \pi_{\kappa,\theta}(a|o') \sum_{o''} \gamma p(o''|a,b') \nabla_\vartheta \tilde{Q}_\kappa(b'',\kappa).\end{aligned}$$

Здесь также замечаем наличие рекурсии, и использование структуры расширенного марковского процесса принятия решений приводит к теореме 2.

Теорема 2 (о градиенте генератора подцелей). *Для фиксированного множества марковских умений со стохастической реализующей стратегией, дифференцируемой по параметрам ϑ, градиент ожидаемой дисконтированной отдачи по параметрам ϑ с начальными условиями (b_1,κ_0) равен*

$$\nabla_\vartheta \tilde{Q}_\kappa(b,\kappa) = \sum_{b',\kappa} \mu_\kappa(b',\kappa|b_1,\kappa_0) \nabla_\vartheta \beta_{\kappa,\vartheta}(o')(V_\kappa(b') - Q_\kappa(b',\kappa)), \tag{11}$$

где $\mu_\kappa(b', \kappa | b_1, \kappa_0)$ *– дисконтированные частоты появления предполагаемого состояния и умения по траекториям, начинающимся с начальных условий* (b_1, κ_0).

Таким образом, сформулированы две теоремы, которые позволяют получить выражения для обновления набора параметров в процедуре SLAPLearn.

6. Возможности применения подхода SLAP в управлении беспилотным транспортным средством

Предложенная архитектура SLAP агента может служить теоретическим обоснованием реализации адаптивной системы управления беспилотным автомобилем. На рис. 4 представлена схема интеграции SLAP агента в широко распространенную в индустрии систему управления беспилотными автомобилями Apollo. В данном случае предполагается, что стандартный модуль планирования будет заменен модулем, который помимо планирования позволяет сохранять опыт, дообучаться и использовать настроенные стратегии для улучшения и ускорения этапа планирования. Таким образом, наряду с рядом подсистем, которые используются в Apollo и в STRL (локализации, построение карты и т.д.), за адаптивное планирование поведения здесь отвечает SLAP агент. В системе Apollo интеграция всех этих модулей осуществляется на основе модифицированной робототехнической операционной системы (Apollo Cyber RT и системы реального времени RTOS), которая позволяет обмениваться сообщениями в асинхронном режиме различным подсистемам.

Рассмотрим задачу обгона автомобилем динамических препятствий (других автомобилей) на многополосном шоссе (рис. 5). Модуль построения карты выдает информацию о границах дороги, полосах и разрешенных скоростях. Модуль локализации позволяет агенту определить свое положение и скорости на шоссе. Модуль построения сенсорной модели передает информацию о движущихся объекта и их скоростях. Модуль предсказания траектории выдает предполагаемые траектории всех объектов на сцене с некоторым горизонтом планирования. В модуле одновременного планирования и обучения реализуется схема по иерархическому адаптивному планированию. На верхнем уровне составляется абстрактный план по действиям, которые должен совершить агент (перестроение направо, перестроение налево). Данный план

Рис. 4. Схема использования SLAP агента в качестве модуля в системе управления Apollo.

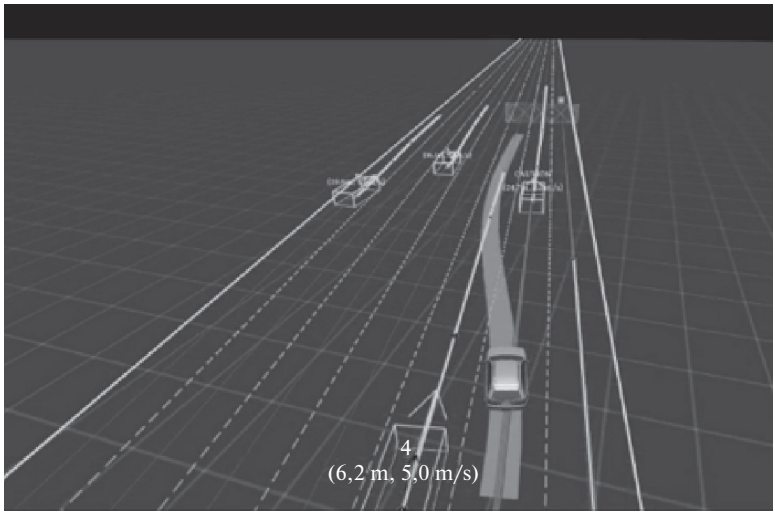

Рис. 5. Пример реализации сценария обгона динамических препятствий в системе управления Apollo с использованием предложенной концепции адаптивного планирования. Светлые тонкие линии — предсказываемые траектории объектов, светлая полоса — планируемая траектория агента.

строится путем поиска кратчайшего на графе поведенческого дерева (вычислительный аналог дерева Монте-Карло в Apollo), в котором реализованы основные правила и условиях обгона (см. подробности реализации в [17]).

Каждый шаг высокоуровневого плана уточняется в процессе обучения (реализации построенных планов) в симуляторе, и каждая часть общего маневра обгона настраивается для субоптимальной реализации соответствующей части общей траектории с использованием обучения с подкреплением или с его "мягкой версией" — эволюционным программированием. Наконец, сглаживание построенных траекторий с учетом динамики объекта управления происходит в модуле управления движением.

Интеграция в общую схему управления автомобилем, реализуемую в Apollo, подсистемы SLAP позволяет добиться большей адаптивности получаемых решений и позволяет автоматизировать процесс задания условий для совершения безопасного маневра.

7. Заключение

В статье представлен новый подход к задаче интеграции подсистем планирования и обучения в иерархических системах управления мобильными робототехническими платформами и беспилотными транспортными средствами. Была предложена оригинальная иерархическая постановка задачи одновременного планирования и обучения, в которой предлагается провести двойную декомпозицию задачи. Первая декомпозиция реализуется за счет выделения абстрактных действий — умений и обучаемых стратегий, которые реализу-

ют их на операционном уровне. Вторая декомпозиция касается выделения предметной среды в ситуации и упрощении модели, используемой для обучения актора, формирующего стратегию, и критика, оценивающего полезность ситуаций. В статье предложены два примера, в которых продемонстрированы особенности работы архитектуры SLAP агента, решающего поставленную задачу: модельная задача с комнатами и важная индустриальная задача планирования маневров беспилотного автомобиля. Предложенные принципы обучения SLAP агента теоретически обоснованы.

В дальнейшем предполагается развитие предложенного подхода в двух направлениях. Первое направление предполагает теоретическое исследований сложностных свойств предложенных алгоритмов, критериев сходимости процесса обучения и нижних гарантируемых оценок качества получаемых стратегий агента. Второе направление предполагает продолжение проработки принципов обновления и использования объектной модели сенсорной ситуации для более эффективной работы процедур планирования.

СПИСОК ЛИТЕРАТУРЫ

1. *Trafton G.J., et al.* ACT-R/E: An Embodied Cognitive Architecture for Human-Robot Interaction // J. Human-Robot Interaction. 2013. V. 2. No. 1. P. 30–54.
2. *Goertzel B.* From Abstract Agents Models to Real-World AGI Architectures: Bridging the Gap // Lecture Notes in Computer Science / ed. Everitt T., Goertzel B., Potapov A. Cham: Springer International Publishing, 2017. V. 10414. P. 3–12.
3. *Wu J., et al.* Track to Detect and Segment: An Online Multi-Object Tracker // 2021 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR). IEEE, 2021. P. 12347–12356.
4. *Likhachev M., Ferguson D.* Planning long dynamically feasible maneuvers for autonomous vehicles // Int. J. Robotics Research. 2009. V. 28. No. 8. P. 933–945.
5. *Aitygulov E., Kiselev G., Panov A.I.* Task and Spatial Planning by the Cognitive Agent with Human-like Knowledge Representation // Interactive Collaborative Robotics. ICR 2018. Lecture Notes in Computer Science / ed. Ronzhin A., Rigoll G., Meshcheryakov R. Springer, 2018. V. 11097. P. 1–12.
6. *Саттон Р.С., Барто Э.Г.* Обучение с подкреплением. М.: БИНОМ. Лаборатория знаний, 2011. Изд. 2-е.
7. *Moerland T.M., Broekens J., Jonker C.M.* Model-based Reinforcement Learning: A Survey. 2020. P. 421–429.
8. *Макаров Д.А., Панов А.И., Яковлев К.С.* Архитектура многоуровневой интеллектуальной системы управления беспилотными летательными аппаратами // Искусственный интеллект и принятие решений. 2015. № 3. С. 18–33.
9. *Yakovlev K., et al.* Combining Safe Interval Path Planning and Constrained Path Following Control: Preliminary Results // Interactive Collaborative Robotics. ICR 2019. Lecture Notes in Computer Science. 2019. V. 11659. P. 310–319.
10. *Staroverov A., et al.* Real-Time Object Navigation with Deep Neural Networks and Hierarchical Reinforcement Learning // IEEE Access. 2020. V. 8. P. 195608–195621.
11. *Киселев Г.А.* Интеллектуальная система планирования поведения коалиции робототехнических агентов с STRL архитектурой // Информационные технологии и вычислительные системы. 2020. № 2. С. 21–37.

12. *Pack L., Littman M.L., Cassandra A.R.* Planning and acting in partially observable stochastic domains // Artificial Intelligence. 1998. V. 101. P. 99–134.
13. *Bacon P.-L., Harb J., Precup D.* The Option-Critic Architecture // Proc. of the AAAI Conf. on Artificial Intelligence. 2017. V. 31.
14. *Keramati R., et al.* Strategic Object Oriented Reinforcement Learning. 2018.
15. *Watters N., et al.* COBRA: Data-Efficient Model-Based RL through Unsupervised Object Discovery and Curiosity-Driven Exploration. 2019.
16. *Hafner D., et al.* Dream to Control: Learning Behaviors by Latent Imagination // Int. Conf. on Learning Representations. 2020.
17. *Jamal M., Panov A.* Adaptive Maneuver Planning for Autonomous Vehicles Using Behavior Tree on Apollo Platform // Artificial Intelligence XXXVIII. SGAI 2021. Lecture Notes in Computer Science / ed. Bramer M., Ellis R. 2021. V. 13101. P. 327–340.

Статья представлена к публикации членом редколлегии О.П. Кузнецовым.

Поступила в редакцию 31.10.2021

После доработки 09.01.2022

Принята к публикации 26.01.2022