Содержание

1 Введение **3**

2 Формальная постановка задачи **5**

3 Обзор литературы **6**

3.1 Обучение функций расстояния 7

3.2 Введение в используемые методы 12

3.2.1 Dreamer 12

3.2.2 PEG 14

3.2.3 Преемственные признаки 17

4 Функции расстояния в моделях мира **19**

4.1 Анализ признаковых описаний состояния 19

4.2 Локальное вознаграждение 22

4.3 Динамическое расстояние, основанное на приблизительном динамическом программировании 23

5 Вычислительные эксперименты **26**

5.1 Точечный лабиринт 27

5.1.1 Использование аналитических расстояний и преемственных представлений 27

5.1.2 Анализ динамических расстояний на точечном лабиринте 30

5.1.3 Анализ динамических расстояний на среде со сложным управлением 33

6 Заключение **37**

Аннотация

Задачей целенаправленного обучения с подкреплением является разработка агентов, способных достигать произвольных целей в среде. Из-за большого разнообразия целей, задание вознагражденииия вручную может быть или трудной, или вовсе невозможной задачей. Поэтому, необходимо разрабатывать алгоритмы, способные обучаться без внешнезаданной функции вознаграждения. Такой подход к задаче обучения с подкреплением может быть формализован как задача целенаправленного обучения с подкреплением без учителя.

Для решения этой задачи, используются обучаемые методы подсчёта вознаграждения. В данной работе мы рассматриваем семейство таких вознаграждений, основанных на функции расстояния, и анализируем их поведение в совокупности с моделями мира и различными репрезентациями, на которых эти расстояния обучаются. Для проверки наших предположений, мы проводим эксперименты на среде с простым управлением и сложной конфигурацией лабиринта: точечный лабиринт, и на среде со сложным управлением, но простой топологией лабиринта, используя в качестве агента модель муравья. В среде точечного лабиринта нам удалось продемонстрировать улучшение в достижении далёких целей, в то время как в среде с муравьем наш метод показал результаты, сопоставимые с оригинальным методом.

1 Введение

Человечество, на протяжении всего своего существования, увеличивало производительность труда. Это необходимо как для удовлетворения материальных потребностей человека, так и для повышения качества жизни и отдачи от любой деятельности: затрачивается меньше времени на необходимые, но при этом рутинные задачи. Следующим логичным шагом, после разработки конвееров и устройств, способных высокоэффективно выполнять элементарные действия, является разработка агентов, способных в реальной среде выполнять широкий спектр рутинных задач. Подобная задача отлично формализуется с помощью целенаправленного обучения с подкреплением: в зависимости от поставленной задачи, агент выполняет тот или иной набор действий, необходимый для успешного выполнения задачи. Важно заметить, что спектр рутинных задач довольно широк, и, фактически, обозначает что мы ожидаем от агента умения достигать произвольных состояний в среде. Такая постановка задачи порождает целый спектр проблем:

1. В случае, если мы хотим обучить агента достижению произвольных целей в среде, нам необходимо ставить цели сопоставимо текущему навыку агента: невозможно пройти из комнаты на кухню, если ты ещё не умеешь ходить.

2. Для обучения агента достижению произвольных целей в среде необходима функция вознаграждения, которая будет индуцировать поведение агента. Однако в реальной среде, в случае, если мы хотим научиться достигать произвольных целей, невозможно создать такую функцию вознаграждения: нам придётся снабжать датчиками все предметы, все точки среды, в которой происходит обучение, чтобы грамотно отслеживать факт достижения тех или иных состояний.

3. Взаимодействие со средой - времязатратный процесс: на выполнение одного действия могут уходить десятки или сотни милисекунд (в силу необходимости управлением сочленениями). Это сильно замедляет скорость обучения агента. В купе с множеством целей, которые агенту необходимо обучиться достигать, обучение может занимать непозволительно длительное время.

Первую проблему можно решить, выбирая цели 'средней' сложности. Фактически, это цели, достижение которых доставляет трудности агенту, но при этом он способен их достичь. Понятно, что поставив слишком сложную цель агенту, он впринципе не

сможет её достичь. В случае, если цель будет слишком простой, агент не научится ничему новому: он и так умел достигать эту цель.

Для решения второй проблемы предлагается создание процедуры обучения функции вознаграждения, которая и будет определять успешность достижения различных целей.

У третьей проблемы два решения. Первое - обучение в высококачественнных симуляторах, точно отражающих динамику среды, в которой планируется использование агента. Второй же путь, более общий, предполагает использование моделей мира: аппроксимаций, способных предсказывать отклик реальной среды на действие агента. Для того, чтобы модель мира соответствовала действительности, помимо целенаправленной стратегии, обучаемой достижению целей, должна присутствовать и исследовательская стратегия, способная генерировать новые способы взаимодействия со средой, чтобы модель мира могла как можно более точно соответствовать действительности.

Один из основополагающих подходов, решающих проблему достижения произвольных целей в среде, является подход Go-explore [12]. Основная идея заключается в том, что мы можем выбирать состояния из числа посещенных, каким-то образом достигать их и затем исследовать из них (с помощью, например, случайной стратегии). Сама работа является прототипом, поскольку достижение состояния, из которого будет проводится исследование, достигается программными путями (сохранением состояния среды в специальном буфере). Но на текущий момент некоторые работы реализуют данный подход без использования программных методов: Cell-free latent go-explore [21] и PEG [22].

В данной работе мы используем алгоритм PEG, поскольку он решает все три проблемы:

1. Постановка целей осуществляется таким образом, что максимизируется исследование, проводимое из последнего достигнутого состояния, при достижении текущей цели. Таким образом, учитывается и прогресс в достижении цели, и новизна цели (старые цели будут обеспечивать малый прогресс в исследовании).

2. В качестве функции вознаграждения для обучения целенаправленной стратегии используется обучаемая функция расстояния между состояниями. Обучение происходит на траекториях, генерируемых стратегией.

3. Для увеличения числа взаимодействий в единице времени используется модель мира Dreamer [16].

Мы обращаем наше внимание на функцию вознаграждения, исследуя два вопроса: какие необходимо использовать представления для обучения модели, и как именно обучать функцию расстояния. Мы рассматриваем представления, генерируемые непосредственно из наблюдений, и представления состояний среды, генерируемые алгоритмом Dreamer [16]. В качестве рассматриваемых функций расстояния, мы изучаем два метода.

Первый метод, используемый в алгоритме [22], предлагает обучать функцию расстояния между состояниями с помощью решения задачи регрессии. В случае, если дана некоторая траектория, то нам известно расстояние между любой парой состояний, которое понадобилось преодолеть стратегии, чтобы перейти из одного состояния траектории в другое. Тогда, целевой переменной для данной пары состояний будет являться количество действий, совершенное, чтобы попасть из одного состояния в другое.

Второй метод заключается в обучении функции расстояния с помощью динамического программирования: расстояние от текущего состояния до целевого на 1 меньше расстояния от следующего состояния до целевого. Новизной данной работы является применение расстояния, обучаемого с помощью динамического программирования, в моделях мира, анализ вышеописанных представлений в моделях мира, и решение проблем, связанных с применением представлений состояний среды, генерируемых алгоритмом Dreamer [16]. Основными задачами являются реализация вышеописанных идей, их анализ и устранение возникающих недостатков.

2 Формальная постановка задачи

Рассмотрим множество $(S, A, G, R, T, \rho_0, \gamma)$, где:

- S - множество состояний среды
- A - множество допустимых действий в среде
- G - множество целей среды. В нашей постановке, $G \subset S$
- R - функция вознаграждения: $r(s, a, s') \to \mathcal{R}$

- T - функция переходов: $S \times A \to P$, где P - множество функций распределения на S.
- p_{s_0} - функция распределения начальных состояний
- $\gamma \in [0; 1]$ - дисконтирующий множитель

Целенаправленной стратегией называется параметризованная функция, выдающая функцию распределения на множестве действий: $\pi_{goal}(a|s, g)$ при условии нахождения в состоянии s и поставленной цели g. Стратегия вместе с функцией переходов, распределением целей p_g и начальных состояний p_{s^0} задает распределение действий и состояний в зависимости от временного шага t:

$$p(s^i) = \mathop{\mathbf{E}}_{s^0 \sim p_{s^0}(\cdot), g \sim p_g(\cdot)} \prod_{k=0}^{i-1} T(s^{k+1}|s^k, a^k)\pi_{goal}(a^k|s^k, g)$$

$$p(s^i|s^j) = \mathop{\mathbf{E}}_{g \sim p_g(\cdot)} \prod_{k=j}^{i-1} T(s^{k+1}|s^k, a^k)\pi_{goal}(a^k|s^k, g)$$

$$p(a^i) = \mathop{\mathbf{E}}_{s^0 \sim p_{s^0}(\cdot), g \sim p_g(\cdot)} \pi_{goal}(a^i|s^i, g) \prod_{k=0}^{i-1} T(s^{k+1}|s^k, a^k)\pi_{goal}(a^k|s^k, g)$$

$$p(a^i|s^i) = \mathop{\mathbf{E}}_{g \sim p_g(\cdot)} \pi_{goal}(a^i|s^i, g)$$

Тогда, задачей агента является максимизация ожидаемой отдачи:

$$\mathop{\mathbf{E}}_{s^i \sim p(s^i), a^i \sim p(a^i)} \sum_{i=0}^{Term} \gamma^i r(s^i, a^i, s^{i+1}),$$

где $Term$ - момент окончания эпизода.

При решении нашей задачи обучению достижения произвольных целей в среде, функция вознаграждения отсутствует. Поэтому, нашей задачей также является разработка такой функции вознаграждения, которая будет индуцировать желаемое поведение агента, а именно - достижение произвольных целей в среде.

3 Обзор литературы

В данной работе основное внимание уделяется функциям расстояния. Здесь мы приводим работы, которые так или иначе связаны с обучением функций расстояния, или с их непосредственным применением.

3.1 Обучение функций расстояния

Есть несколько работ, изучающих обучение функций расстояния для целенаправленного обучения с подкреплением. Например, в PEG [22] используется метод обучения функций расстояния, представленный в статье DDL [10]. Процедура обучения функции расстояния (в дальнейшем, динамическое расстояние) заключается в оптимизации следующего функционала:

$$\mathop{\mathbf{E}}_{i\sim U[0;Term],j\sim U[i;Term],s^i\sim p(s^i),s^j\sim p(s^j|s^i)} (dd_\psi(s^i,s^j)-|i-j|)^2,$$

где dd - функция динамического расстояния, ψ - параметры нейронной сети, $U[a;b]$ - равномерное распределение на отрезке $[a;b]$. Фактически, этот функционал обучает предсказывать расстояние между двумя состояниями траектории $s^i, s^j \in \tau = (s^0, a^0, \ldots, s^i, a^i, \ldots, s^j, a^j, \ldots, s^{Term})$, порождённой $a^i \sim \pi_{goal}(a^i|s^i, g)$. То есть, обучение функции расстояния происходит с помощью решения задачи регрессии на парах состояний траектории.

Авторы статьи DDL [10] показывают, что при совместном обучении стратегии и функции расстояния, функция расстояния будет предсказывать кратчайшее расстояние, а стратегия будет оптимальной:

Теорема 3.1. *В случае, если динамика среды детерминирована, для любой пары состояние-цель (s, g) функция расстояния сойдётся к кратчайшей функции расстояния, а функция стратегии - к оптимальной стратегиию*

Доказательство приведено в оригинальной статье DDL [10].

Есть и другие способы обучения функций расстояния. Так, например, авторы Learned action distance [15] предлагают для подсчёта расстояния использовать не функцию расстояния, а норму разности между репрезентациями состояний. Соответственно, и задача обучения заключается в формировании репрезентации состояний. Однако проблема в таком подходе заключается в том, что расстояние (измеряемое в количестве действий), между парой состояний, не симметрично. Так, например, если мы плывём на лодке по сильному течению, чтобы спустится вниз, достаточно сделать пару гребков. Однако чтобы подняться вверх, необходимо большее число гребков.

Чтобы избавиться от этой проблемы, предлагается обучать 'среднее' расстояние: среднее количество действий, которое необходимо совершить, чтобы попасть из

текущего состояния в целевое и обратно.

$$d(s^i, s^j) = \frac{m(s^i|s^j)}{2} + \frac{m(s^j|s^i)}{2},$$

где $m(s^i|s^j)$ - количество дейтсвий, которое необходимо совершить агенту, чтобы из состояния s^j достичь состояния s^i.

Проблема в таком подходе заключается в том, что во время обучения стратегии достижению целей мы чаще всего просто ставим задачу достижения целевого состояния; при этом, задачи достижения из целевого состония текущего не стоит. Поэтому, авторы взвешивают расстояние от текущего состояния до целевого и расстояние от целевого состояния до текущего пропорционально количеству траекторий, которые содержали эти цели в себе как начальную и конечную точки траекторий соответственно.

$$d(s^i, s^j) = \frac{\rho(s^i)m(s^j|s^i)}{\rho(s^i) + \rho(s^j)} + \frac{\rho(s^j)m(s^i|s^j)}{\rho(s^i) + \rho(s^j)},$$

где $\rho(s^i)$ - вероятность нахождения в состоянии s^i. (Точнее: В случае, если задан Марковский процесс принятия решений, а стратегия зафиксирована, то комбинация марковского процесса принятия решений с зафиксированной стратегией образует марковскую цепь. В этом случае, $\rho(s^i)$ - значение стационарного распределения в состоянии s^i, в случае, если оно определено). Это упрощает задачу, и позволяет авторам реализовать их метод на практике. В дальнейшем, представления состояний обучаются таким образом, чтобы норма разности между ними была равна 'среднему' расстоянию.

Статья Neural all pairs [19] развивает идею обучения функции расстояния с помощью решения задачи регрессии [10], предлагая целый ряд улучшений. Во-первых, заметим что для обучения динамического расстояния требуются траектории. С одной стороны, ничто не мешает использовать траектории, полученные текущей стратегией. Однако тогда мы игнорируем прошлый опыт взаимодействия со средой: среди него могут содержаться полезные переходы, или редкие. Поэтому, полезно также использовать переходы, сгенерированные на предыдущих итерациях оптимизации стратегии. С другой стороны, использование предыдущего опыта также полезно с точки зрения скорости обучения: для совершения одного градиентного шага нам достаточно выбрать траектории из прошлого опыта, а не собирать траектории заново. Для использования траекторий, сгенерированных на прошлых итерациях оптимизации стратегии, в функцию расстояния вводится обуславливание на совершенное действие: $dd_\psi(s^i, a^i, g)$. Чтобы избежать проблем с тем, что использование прошлого

опыта будет предсказывать завышенные оценки расстояния для текущей стратегии (старая стратегия могла достигать целей не так хорошо, как новая), предлагается игнорировать предсказание расстояния по старой траектории, если текущая оценка расстояния 'достаточно' уверенная. В случае, если мы никогда не достигали цели g из состояния s, можно использовать верхнюю оценку расстояния через неравенство треугольника. Пусть у нас имеется некоторое состояние $s_{intermediate}$, которое мы достигали из пары состояние-действие (s, a), и из которого достигалось целевое состояние g. Тогда, верхняя оценка расстояния для пары состояние-дейсвтие (s, a) равна:

$$d(s, a, g) = d(s, a, s_{intermediate}) + \min_{a} \; d(s_{intermediate}, a, g).$$

Такая оценка позволяет ускорить сходимость расстояния на недостигнутых ранее тройках состояние-действие-цель.

В случае, если у нас имеется модель частоты посещений данной пары состояние-действие $N(s^i, a^i)$, можно посчитать степень доверия предсказанным $dd_\psi(s^i, a^i, g)$ расстояниям как:

$$U(s^i, a^i, g) = \frac{C}{(1 + N(s^i, a^i))^M},$$

где C, M - гиперпараметры. Тогда, оптимизируемые функционалы имеют вид:

$$L_{temporal} = \mathop{\mathbf{E}}_{\substack{i,j\sim U[0;Term] \\ s^i\sim p(s^i), a^i\sim p(a^i|s^i) \\ s^j\sim p(s^j|s^i)}} (dd_\psi(s^i, a^i, s^j) - min(j - i, dd_{\psi_{old}}(s^i, a^i, s^j) + U(s^i, a^i, s^j))^2$$

$$L_{triangular} = \mathop{\mathbf{E}}_{\substack{g\sim G, (s^i,a^i)\sim\mu(g) \\ (s^j,a^j)\sim\overline{\mu}(g)}} (dd_\psi(s^i, a^i, g) - (dd_{\psi_{old}}(s^i, a^i, s^j) + dd_{\psi_{old}}(s^j, a^j, g))))^2, \tag{3.1}$$

где $dd_{\psi_{old}}$ - целевая функция расстояния, с весами обновляемыми медленно (для менее стохастичных оценок целевой переменной, впервые появилась в [4]), $\mu(g) = (s, a) : (s, a) \in D \wedge (N(s, a, s_g) = 0)$, D - память прецендентов. Оператор минимизации в функции потерь $L_{temporal}$ используется для учёта оценок функции расстояния в случае его высокой уверенности в существовании более низкого расстояния засчёт применения $U(s^i, a^i, g)$. $L_{triangular}$ обновляет функцию расстояния для тех троек состояние-действие-цель (s, a, g), для которых не было ни одного прецендента достижения цели. Таким образом, в случае отсутствия примера достижения цели, используется верхняя оценка для расстояния по неравенству треугольника.

Все эти методы хорошо работают, если функция динамики среды не случайна. Однако в случае присутствия случайности в динамике, эти методы могут приводить

к вырожденным решениям. Рассмотрим, например, следующий модельный пример на Рис. 1.

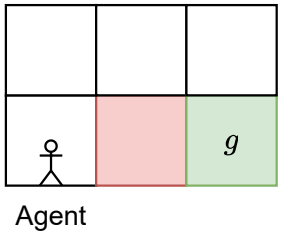

Рис. 1: Клеточный мир. Зелёным цветом обозначено целевое состояние, красным - стохастичное терминальное. В начале эпизода агент всегда оказывается в клетке с изображением агента.

Пусть, при попадании в красное состояние эпизод заканчивается с вероятностью $p = 0.9$, а с вероятностью $p = 0.1$ нет. Все белые состояния с вероятностью $p = 1$ не приводят к окончанию эпизода. В случае долгого обучения, в памяти прецендентов окажутся как траектории, пролегающие через красное состояние, так и траектории, не пролегающие через него. Тогда, в памяти прецендентов найдётся траектория, пролегающая из начального состояние в целевое через красное состояние. Расстояние от красного состояния до целевого - 1. Если мы будем использовать это расстояние как вознаграждение (напр. $r(s, a, s'|g) = -dd_\psi(s'|g)$), то это будет одна из двух клеток, в которой достигается высокий уровень вознаграждения, что будет поощрять агента прохождению через красное состояние. Однако мы бы могли предпочитать гарантированное, но медленное достижение зелёного состояния быстрому, но не гарантированному. Для решения этой проблемы, в статье Distributional NCE [23] предлагается использовать не детерминированный, а стохастичный подход к определению функции расстояния.

Иной способ обучения функций расстояния заключается в применении идей динамического программирования. Так, например, одной из первых работ, предложивших обучение таких функций расстояния является работа Learning to achieve goals [2]. В случае, если нам задана функция вознаграждения, равная -1, и определён критерий достижения цели, то функция полезности, обученная на таком вознаграждении, будет предсказывать расстояние, необходимое агенту чтобы достичь цели.

Развитие этой идеи произошло в работе Multi-task Reinforcement Learning with a Planning Quasi-Metric [17]. Аналогично предыдущим работам, вводится обуславливание на действие для обучения на данных, не порождённых текущей стратегией. Для дополнительной стабильности обучения также вводится обучение на парах последовательных состояний. Таким образом, оптимизируемый функционал имеет вид:

$$\mathop{\mathbf{E}}_{\substack{i,j\sim U[0;Term],i<j \\ s^i\sim p(s^i),a^i\sim p(a^i|s^i),s^j\sim p(s^j|s^i)}} (dd_\psi(s^i,a^i,s^{i+1})-1)^2+(dd_\psi(s^i,a^i,g)-(1+max_a dd_\psi(s^{i+1},a,g)))^2. \tag{3.2}$$

Первое слагаемое в уравнении 3.2 отвечает за обучение расстояния на парах последовательных состояний, в то время как второе - за обучение расстояния аналогично уравнению Беллмана.

Есть и другие работы, посвященные функциям расстояния. Так, например, в работе search on the replay buffer [11], предлагается строить ориентированный граф по памяти прецендентов. Правило проведения ребра следующее: два состояния соединены ребром тогда и только тогда, когда из одного состояния в другое можно попасть за небольшое число шагов. Тогда, путь из начального состояния до целевого можно найти с помощью любого алгоритма поиска пути на графе (напр., алгоритм Дейкстры). Может возникнуть вопрос: зачем искать путь на графе, когда можно просто обучать стратегию достижению любого целевого состояния из начального? Как отмечают авторы, алгоритмы целенаправленного обучения с подкреплением плохо справляются с достижением состояний, находящихся далеко от текущего. Для решения этой проблемы, они предлагают перемещаться по рёбрам с помощью целенаправленной стратегии, т.е. каждый раз, когда для достижения цели необходимо переместиться по ребру, авторы передают агенту в качестве цели конечное состояние на ребре. Тогда задача достижения цели разбивается на задачи перемещения по рёбрам графа.

С другой стороны, можно попытаться выучить функцию расстояния, которая не будет зависеть от стратегии вообще. В дальнейшем, такую функцию расстояния можно использовать как меру близости состояний в среде. Тогда, у нас появляется некоторый критерий, который позволяет различать состояния между собой, и, как следствие, мы сможем создать буфер разнообразных целей для обучения агента. Такой подход и реализует статья walk the random walk [20].

Также интересно отметить, что и авторы DDL [10] предлагают использовать свою функцию расстояния не только для достижения целей. Идея заключается в том, что если выбирать в качестве целей достаточно удалённые состояния, то можно

побуждать агента исследовать среду: в качестве цели может быть выбрана как близкая (в пространстве состояний) цель, поведение на которой агентом ещё не освоена, так и далёкая, приводящая к исследованию среды.

Такие функции расстояния могут использоваться и при обучении на неоптимальных демонстрациях, как было показано в работе HRLfD-RbRS [24]. Пусть у нас задана некоторая неоптимальная демонстрация. Тогда, поставив подцели в некоторой окрестности точек (окрестность определяется как расстояние (в действиях) от состояния) продемонстрированной траектории, можно найти более оптимальную траекторию.

3.2 Введение в используемые методы

В данной работе исследуется поведение функций расстояния, обучаемых на состояниях, генерируемых моделью мира. Данная работа является доработкой функции расстояния, используемой в алгоритме PEG [22]. Поэтому является необходимым описать вышеописанные алгоритмы для понимания работы.

3.2.1 Dreamer

Dreamer [16] - алгоритм обучения с подкреплением, основанный на модели. Dreamer [16] использует модель мира, впервые введённую в методе PlaNet [14]. Основная идея заключается в использовании априорной и апостериорной моделей мира:

$$p_{prior}(s^t|s^{t-1}, a^{t-1}), p_{post}(s^t|s^{t-1}, a^{t-1}, o^t),$$

где s^t состоит из детерминированной - h^t и стохастичной - z^t частей и получается их конкатенацией. Априорная модель мира может быть использована для моделирования функции переходов среды (Такое использование также называется 'воображением', и будет обозначаться как $\hat{T}(s^t|s^{t-1}, a^{t-1})$ - аппроксимация модели переходов.) Апостериорная модель мира используется для обучения априорной модели мира с помощью минимизации дивергенции Кульбака-Лейблера:

$$D_{KL}(p_{prior}(s^t|s^{t-1}, a^{t-1})||p_{post}(s^t|s^{t-1}, a^{t-1}, o^t))$$

Для обучения апостериорной модели мира используется метод максимального правдоподобия, в котором каждая из плотностей обусловлена на текущее состояние модели мира. Максимизируется значения плотности наблюдения, вознаграждения и

сигнала конца эпизода, т.е. максимизируется следующая функция:

$$L = ln(p(o^t|s^t)) + ln(p(r^t|s^t)) + ln(p(\gamma^t|s^t)),$$

где

$$\gamma^t = \begin{cases} 0, & \text{если эпизод окончен} \\ \gamma, & \text{иначе} \end{cases}$$

Схема метода (за авторством [16]), приведена ниже.

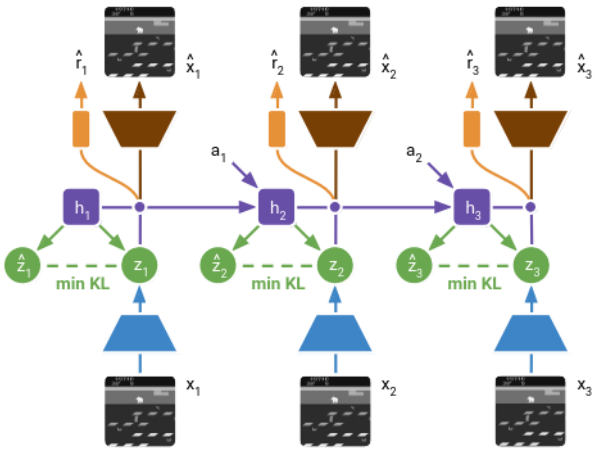

Рис. 2: $\hat{z}$ - априорная стохастичная часть скрытого состояния s. z - апостериорная стохастичная часть скрытого состояния s. h - детерминированное состояние, не обуславливаемое на наблюдение x (в наших обозначениях - o).

Обучение стратегии происходит полностью в воображении. Поскольку у нас есть аппроксимации функций вознаграждения, переходов, и конца эпизода, то, помимо обучения в воображении, мы также можем обучать стратегию и функцию полезности с помощью градиентных методов. Введём:

$$V^k(s^i) = \mathop{\mathbf{E}}_{r^t \sim p(r^t|s^t), s^t \sim p(s^t|s^{t-1}, a^{t-1}), s^0 \sim D, a^t \sim \pi_\varphi(a^t|s^{t-1})} \sum_{t=i}^{i+k} \gamma^{t-i} r^t + \gamma^{k+1} v_\theta(s^k).$$

Тогда, отдача за воображенную траекторию вычисляется по формуле, аналогичной λ-отдаче, т.е.

$$V_\lambda(s^i) = (1-\lambda) \sum_{j=1}^{H} \lambda^{j-1} V^j(s^i),$$

где H - горизонт воображения, т.е. количество моделируемых в воображении переходов, s^0 - произвольное состояние траектории, просэмплированное из памяти прецендентов D (хранит себе траектории, выполненные в среде). Затем, это значение

можно использовать для оптимизации стратегии и функции полезности:

$$\max_{\varphi} \mathop{\mathbf{E}}_{r^t\sim p(r^t|s^t),s^t\sim p(s^t|s^{t-1},a^{t-1}),s^0\sim D,a^t\sim\pi_\varphi(a^t|s^{t-1})} \sum_{k=0}^{H} V_\lambda(s^k)$$

$$\min_{\theta} \mathop{\mathbf{E}}_{r^t\sim p(r^t|s^t),s^t\sim p(s^t|s^{t-1},a^{t-1}),s^0\sim D,a^t\sim\pi_\varphi(a^t|s^{t-1})} \sum_{k=0}^{H} (v_\theta(s^k) - V_\lambda(s^k))^2.$$

Помимо этой, основной, идеи, в алгоритме применяется множество различных оптимизаций, позволяющих стабилизировать или улучшить процедуру обучения как стратегии, так и модели. Выше описана схема обучения алгоритма Dreamer первой версии, на текущей момент вышли версии DreamerV2 [16] и DreamerV3 [25], что могут достигать ещё лучших значений на средах, чем предыдущая версия. В нашей реализации мы пользуемся алгоритмом DreamerV2, поскольку это версия, на которой базируется алгоритм PEG.

3.2.2 PEG

PEG - алгоритм обучения целенаправленного обучения с подкреплением, основанный на модели. Основная идея заключается в использовании подхода Go-Explore [12]: сначала мы достигаем поставленную цель, а затем, исследуем из поставленной цели. С помощью этой процедуры получается, с одной стороны, пополнять буфер с помощью нового опыта, а с другой - ставить новые цели 'среднего' уровня сложности. Обсудим детали обучения алгоритма [22]. На первом этапе, агент выбирает цель, которую необходимо достичь в реальной среде. Эта цель выполняется планированием в воображении с помощью MPPI [9]. Алгоритм представляет из себя итеративный процесс. Идея заключается в том, что при заданном начальном состоянии s_0 (просэмплированном из среды), горизонте воображения H, на очередной итерации мы сэмплруем некоторый набор целей $g_i \sim MPPI_W(G)$, запускаем целенаправленную стратегию в воображении с целью достичь каждой из целей:

$$\tau_i = (s^0, a^0 \sim \pi(a^0|s^0, g_i), \ldots, s^k \sim \hat{T}(s^k|s^{k-1}, a^{k-1}), a^k \sim \pi(a^k|s^k, g_i), \ldots,);\ |\tau| = 2(H{+}1).$$

По достижении конечного состояния траектории используем оценку функции полезности и для определения целей:

$$w_i = V_{expl}(s^H), s^H \in \tau_i.$$

И, используя данные веса, меняем параметры планировщика для сэмплирования из целей для новых итераций из более узкого региона, чем был на предыдущей итерации. Для этого каждому сэмплу цели задётся вес, пропорциональный исследовательскому вознаграждению, и на сгенерированных сэмплах происходит максимизация правдоподобия многомерного нормального распределения $MPPI_W(G)$. Схема метода представлена на рисунке 3.

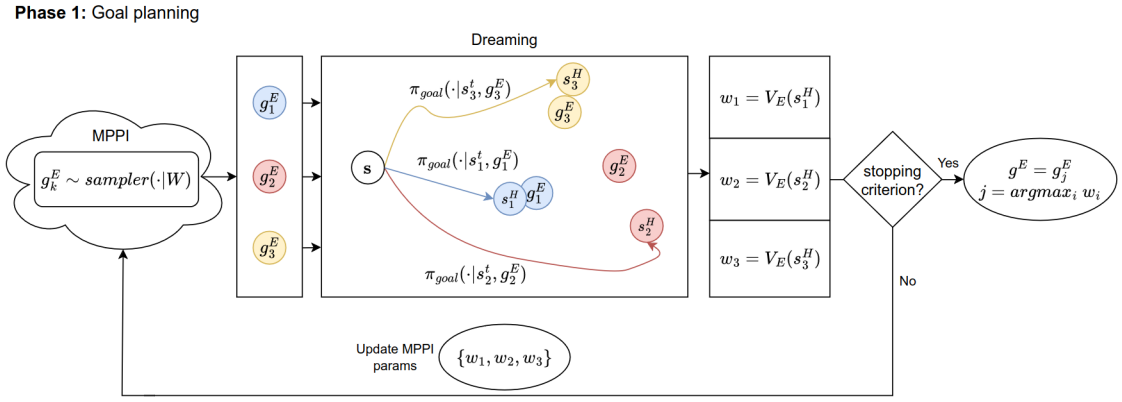

Рис. 3: Первая фаза: Планирование целей выполняется с помощью MPPI. На каждой итерации, сэмплируется набор исследовательских целей g_k^E. Траектории целенаправленной стратегии сгенерированы с помощью целенаправленной стратегии $\pi(\cdot|s, g)$: $\tau_k = (\mathbf{s} = s_k^1, a_k^1, s_k^2, a_k^2, \ldots, s_k^H, a_k^H)$, где $a_k^{t-1} \sim \pi(\cdot|s_k^{t-1}, g_k^E), s_k^t \sim \hat{T}(\cdot|s_k^{t-1}, a_k^{t-1})$. Некоторые цели удаётся достичь в воображении (синим, жёлтым), другие - нет (красным). Веса для дальнейших итераций MPPI расчитываются пропорционально исследовательской отдаче V_E.

После выполнения первой фазы, выбирается некоторая исследовательская цель g^E, достижение которой и будет выполнено в реальной среде. По достижении этой цели, или по превышении числа допустимого количества шагов в среде, целенаправленная стратегия сменяется стратегией исследования. Таким образом, реализуется подход Go-Explore [12]: сначала достигается цель, в которой ожидается высокая исследовательская награда, а затем происходит исследование из этой цели. Схема этого подхода представлена на рисунке 4.

Как можно заметить на рисунке выше, одни и те же состояния поступают как в буфер целей, так и в буфер состояний. Два одинаковых буфера введены для того, чтобы подчеркнуть, что возможно несколько различных методов постановки целей: по батчу, и по среде (оба метода реализованы в коде PEG [22]). На данной картинке, и в дальнейшем, представлен метод постановки целей по батчу. Различие от поста-

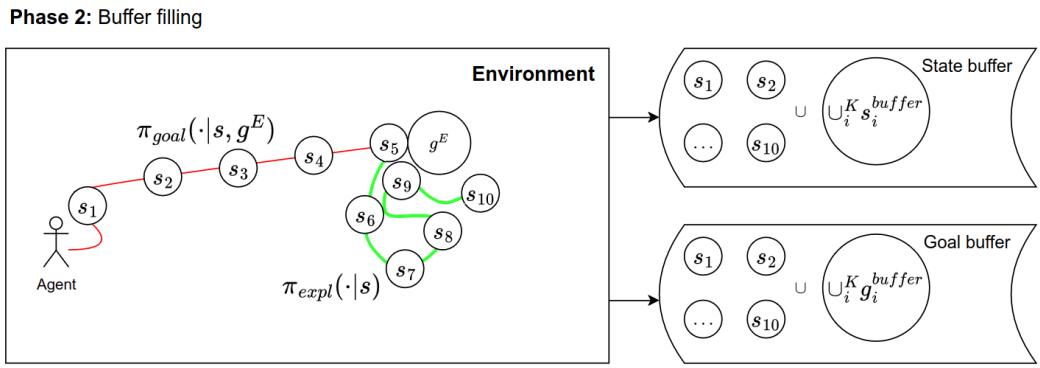

Рис. 4: Вторая фаза: Пополнение буфера. Исследовательская цель g^E, что была выбрана во время планирования целей, ставится в среде. Затем, целенаправленная стратегия $\pi(\cdot|s, g^E)$ выполняется из стартового состояния для достижения целевого. По достижении цели, или при превышении числа шагов в среде, запускается исследовательская стратегия $\pi_{expl}(\cdot|s)$, что исследует среду из конечного состояния.

новки целей по среде заключается в том, что в постановке целей по среде, буфер целей хранит ту цель, что была поставлена в среде, а не ту, что была встречалась в траектории. Как показали наши эксперименты, большой разницы между обоими методами нет.

Параллельно с процедурой пополнения буфера новым опытом, происходит обучение как стратегии, так и функции динамического расстояния. Процедура обучения обоих компонент происходит исключительно в воображении, цели для воображаемых траекторий берутся из буфера целей, как и начальные состояния. Фактически, это обозначает, что воображение происходит из произвольного состояния траектории, что позволяет обучать стратегию на всех состояниях среды, когда либо достигнутых во время обучения. Функция вознаграждения получается из функции расстояния следующим преобразованием: $r(s, a, s'|g) = -dd(s', g)$, т.е. агент обучается минимизировать функцию расстояния до цели. Процедура обучения стратегии соответствует процедуре обучения стратегии в алгоритме Dreamer [16]. Ниже мы приводим подробную схему обучения целенаправленной стратегии и расстояния в воображении (рис. 5).

Процедура обучения функции расстояния заключается в сэмплировании произвольных состояний из траектории. Поскольку нам дана траектория, то нам известно

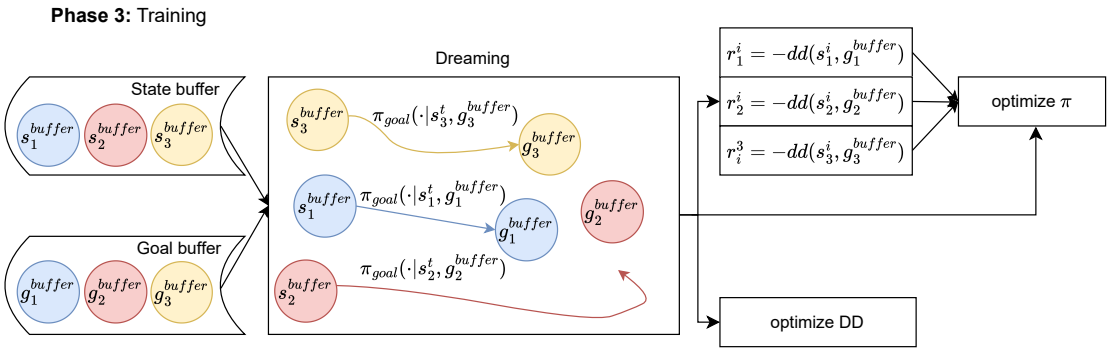

Рис. 5: Третья фаза: обучение стратегии и функции расстояния. Из памяти прецендентов, мы сэмплируем начальные состояния s_k^{buffer} и целевые состояния g_k^{buffer}. Траектории τ получены действием целенаправленной стратегии $\pi_{goal}(\cdot|s,g)$ в воображении: $\tau_k = (s_k^{buffer} = s_k^1, a_k^1, s_k^2, a_k^2, \ldots, s_k^H, a_k^H)$, где $a_k^{t-1} \sim \pi(\cdot|s_k^{t-1}, g_k^{buffer})$, $s_k^t \sim \hat{T}(\cdot|s_k^{t-1}, a_k^{t-1})$. Затем, с помощью функции расстояния, мы подсчитываем вознаграждения за каждый переход. Имея вознаграждение и траектории, можно обучить целенаправленную стратегию. Параллельно, происходит обучение функции расстояния: dd.

расстояние между состояниями этой траектории. Это расстояние и используется в качестве целевой переменной для предсказания. На вход же расстоянию приходит признаковое описание текущего состояния. Говоря формально, имеем следующую процедуру обучения функции расстояния: в случае, если нам задана память прецендентов D, горизонт воображения H, целенаправленная стратегия $\pi_{goal}(\cdot|s,g)$, и функция воображения $\hat{T}(\cdot|s,a)$ (аппроксимация модели переходов среды $T(\cdot|s,a)$), то задача обучения динамического расстояния заключается в оптимизации:

$$
\begin{aligned}
&L_{dd} = \mathop{\mathbf{E}}_{(s^i,s^j)\sim\tau}(dd(s^i,s^j) - |j-i|)^2, \\
&\tau = (s^0, a^0, \ldots, s^i, a^i, \ldots, s^j, a^j, \ldots, s^H, a^H), \\
&s^0, g \sim D; a^t \sim \pi_{goal}(\cdot|s^t, g); s^{t+1} \sim \hat{T}(\cdot|s^t, a^t).
\end{aligned}
\tag{3.3}
$$

Схема обучения функции динамического расстояния представлена на рис. 5., рис. 6.

После обучения стратегий и функций расстояния оценка качества работы целенаправленной стратегии производится с помощью заранее определённого набора целей. (Рис. 7).

3.2.3 Преемственные признаки

На ранних стадиях работы мы проводили эксперименты с преемственными представлениями для обучения функций расстояния. Нам не удалось достичь ка-

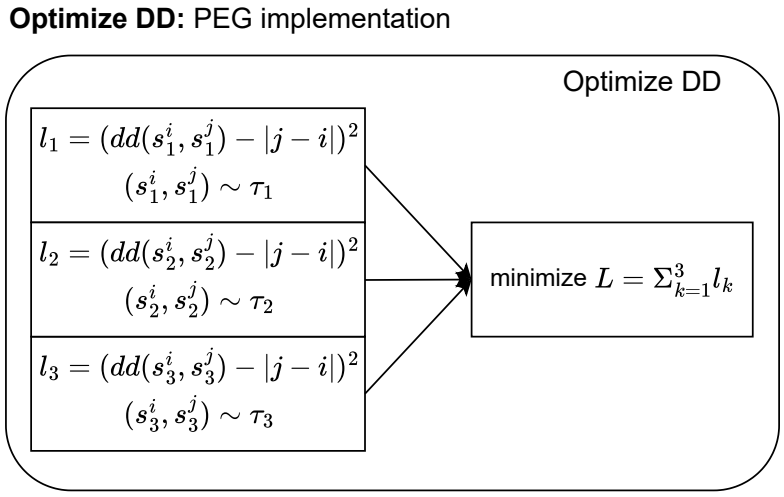

Рис. 6: Имлементация блока оптимизации функции расстояния в алгоритме PEG (для полной картины всей процедуры обучения см. также рис. 5). После воображения траекторий, мы сэмплируем пары состояний из этих траекторий. На парах этих состояний и происходит обучение стратегии.

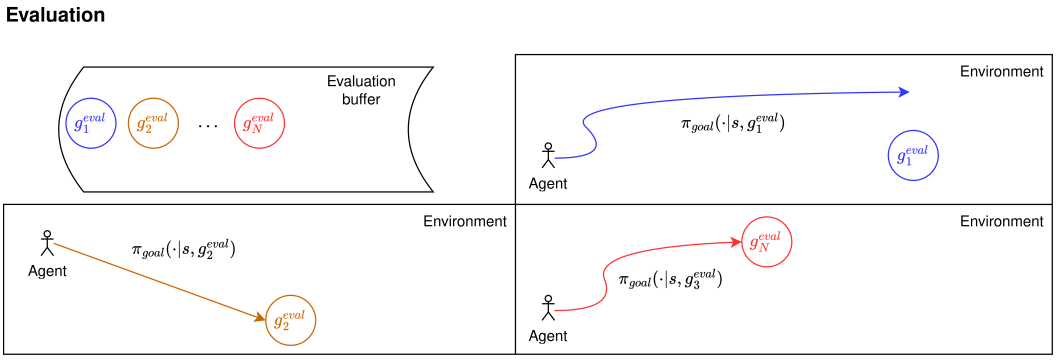

Рис. 7: Иллюстрация процедуры оценки качества работы целенаправленной стратегии: какие-то цели не удаётся достичь (синяя траектория), достижение других не идеально (красная траектория).

чественного поведения функции расстояния с преемственными представлениями, но для полноты картины мы приводим эксперименты с их применением в том числе. Впервые данная идея возникла как преемственные представления в работе successor representations [1]. В случае табличной среды, при решении задачи обучения с подкреплением, состояние среды можно представить в виде One-hot вектора $\vec{s} = (s_0, s_1, \dots, s_i, \dots, s_{m*n})^T$, где i-ая компонента вектора равна единице тогда и только тогда когда агент находится в состоянии s_i, а m и n - широта и высота таблицы соответственно. Тогда, если функция вознаграждения зависит только от конечного состояния среды, т.е. представима в виде вектора: $\vec{r} = (r_0, r_1, \dots, r_i, \dots, r_{m*n})^T$, где r_i

- вознаграждение за попадание в состояние s_i, то отдачу можно записать как:

$$R = \sum_{k=0}^{N} \gamma^k < \vec{s}_k, \vec{r} > = < \sum_{k=0}^{N} \gamma^k \vec{s}_k, r > .$$

Представление $SR(\vec{s}_0) = \sum_{k=0}^{N} \gamma^k \vec{s}_k$ в таком случае называется преемственным представлением состояния $\vec{s}_0$ (N - длина эпизода). Заметим, что в этой записи неявно предполагается наличие последовательности состояний. Эта последовательность состояний, как обычно, сгенерированна агентом при взаимодействии со средой. Т.е., важно заметить, что и как остальные подходы, преемственные представления зависят от поведения агента.

Эту идею можно развить и в случае глубокого обучения с подкреплением. Для реализации этого подхода предлагаются использовать преемственные признаки [6]. Эта идея похожа на идею использования преемственных представлений в том смысле, что преемственные признаки для текущего состояния определяется через сумму (возможно, дисконтированную) будущих преемственных признаков. Формально, пусть у нас имеется состояние s^t и его признаковое описание $e^t \in R^n$, где n - размерность пространства признаковых описаний состояний. Тогда, преемственные признаки состояния $SF(s^t)$ можно записать как:

$$SF(s^t) = \sum_{k=0}^{N} \gamma^k e^{t+k}.$$

Основная идея данного подхода заключается в том, что как и в случае с преемственными представлениями, отдачу можно попробовать аппроксимировать как

$$R = < SF(s^t), w > \tag{3.4}$$

где w - обучаемый вектор весов, предсказывающий отдачу пропорционально сумме каждого из признаков. Несмотря на отсутствие теоретических гарантий возможности аппроксимации произвольной функции полезности с помощью данного подхода, он, тем не менее, показал сравнимый результат с глубоким Q-обучением [4].

4 Функции расстояния в моделях мира

4.1 Анализ признаковых описаний состояния

Введём обозначения, которые нам понадобятся в дальнейшем. Как уже упоминалось ранее, алгоритм [16] может формировать представление состояний s по

имеющейся траектории. Заметим, однако, что это не единственный способ формировать представление состояний. Например, элементарный подход может заключаться в том, что для обучения агента мы будем использовать не полное описание состояния среды s, а лишь признаковое описание наблюдения e. С одной стороны, для полностью наблюдаемых сред, т.е. таких, в которых можно однозначно восстановить состояние среды по поступившему наблюдению, этого подхода достаточно. Однако, представим себе среду в которой задача агента - добраться из одной точки среды в другую, а в качестве входа агенту поступает изображение с камеры. В этом случае, по любому отдельно взятому наблюдению невозможно восстановить состояние среды. Т.е. использование признакового описания наблюдения среды e в одних случаях может замедлять сходимость агента, а в других и вовсе не позволять выучить оптимальную стратегию.

В работах, связанных с применением функций расстояния в моделях мира [18, 22] функцию расстояния предлагается обучать не на признаковом описании состояний, а на признаковом описании наблюдений. В статье LEXA [18] это аргументируется тем фактом, что это позволяет избавится в расстоянии от 'излишней рекуррентной информации'. В обоих статьях предлагается использование динамического расстояния [10] в качестве функции расстояния; однако в LEXA, помимо динамического расстояния также предлагается использование косинусной близости состояний.

Первой идеей является применение, в качестве входа в функцию расстояния, не признакового описания наблюдений $e \in E$, а признакового описания состояний $s \in S$. Такой подход сопряжен с набором нюансов реализации. Так, например, вернёмся к процедуре выбора целей: по батчу, или по среде. Отметим, что планировщик (на рис. 3) планирует цели в пространстве наблюдений O, а не в пространстве E или S. Это позволяет довольно быстро искать необходимые цели, которые могут максимизировать исследовательскую награду в среде. В случае, если мы хотим обучать функцию расстояния на парах признаковых описаний состояний, нам будет необходимо иметь признаковое описание цели.

- В случае постановки целей по батчу, целевое состояние, как и начальное, выбирается как произвольное состояние траектории. Поскольку полная траектория нам доступна, мы можем сформировать признаковое описание как начального, так и целевого состояний траектории.

- В случае постановки целей по среде, целевое состояние для каждой траектории

среды сохраняется вместе со всей траекторией. Но, поскольку, целевое наблюдение было поставлено планировщиком, то у нас нет траектории, которая могла бы породить это целевое наблюдение.

Мы бы могли использовать постановку целей по среде, но при этом обучить кодировщик, который по данному наблюдению бы предсказывал распределение состояний среды, способных сгенерировать это наблюдение. Проблема данного подхода заключается в следующем факте: наблюдение, в некоторых категориях сред, впринципе не может однозначно определить состояние среды. Это значит, что семейство распределений, которое будет порождаться попыткой предсказать зависимость состояния среды от наблюдения, может быть произвольно сложным. Поэтому, мы решили использовать постановку целей по батчу, потому-что представления s для такого метода могут быть легко подсчитаны. На практике, использование признакового описания состояний, а не наблюдений, приводит к ухудшенному локальному достижению целей, но хорошему глобальному достижению целей, т.е. агент приближается к цели, но не может точно её достичь. Это можно заметить из рисунков 8,9.

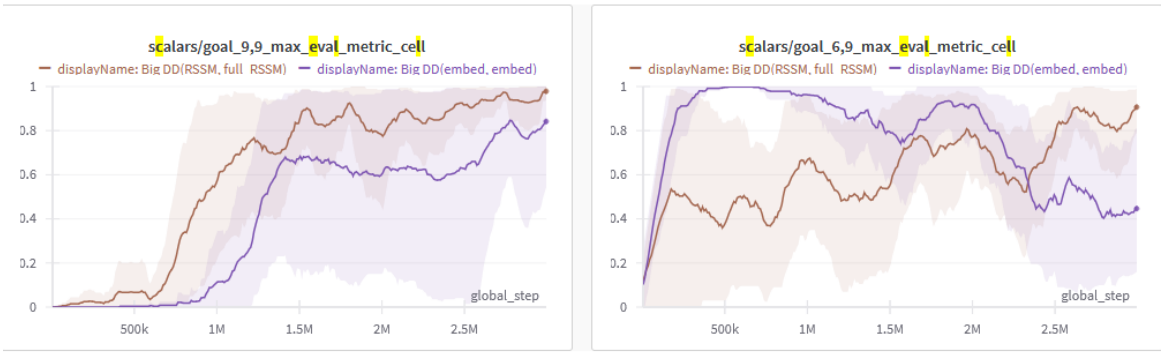

Рис. 8: Иллюстрация успешности достижения двух самых далёких целей в среде: (6, 9) и (9, 9). Фиолетовым и коричневым изображены метрики успешности достижения целей для агента, обученном на расстоянии, принимающем на вход признаковое описание наблюдений и состояний соответственно. Как можно видеть, расстояние, обученное на признаковом описании состояний s, с течением времени обучает агента достижению обоих самых далёких целей среды, в отличии от расстояния, обученного на e.

Из данных рисунков видно, что расстояние, обученное на признаковом описании состояний s позволяет агенту научиться лучше достигать целей (6, 9) и (9, 9), однако

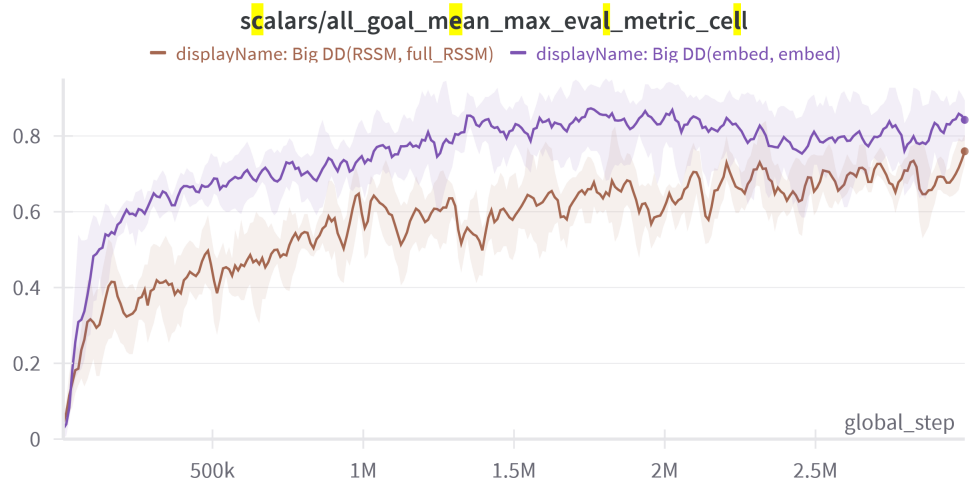

Рис. 9: Иллюстрация средней способности достигать одну из многих целей лабиринта. Фиолетовым и коричневым изображены метрики успешности достижения целей для агента, обученном на расстоянии, принимающем на вход признаковое описание наблюдений и состояний соответственно.

умение достигать всех целей у агента хуже, чем у агента, обученного на вознаграждении, подсчитанном на признаковом описании наблюдений e. Мы же воспринимаем данный результат как положительный, поскольку решить проблему с плохим точным достижением целей можно с введением локального вознаграждения за достижение цели.

4.2 Локальное вознаграждение

Для решения проблемы с точным локальным достижением цели мы предлагаем использовать комбинацию функций вознаграждения: одна будет отвечать за глобальную стратегию достижения цели, а вторая будет помогать агенту точно достигать поставленной цели. В качестве локального вознаграждения мы рассматривали два варианта:

- Предположим, среда находится в состоянии s^i, а перед целенаправленной стратегией $\pi_{goal}(\cdot|s^i, g)$ стоит цель g. По состоянию s^i можно получить распределение действий $a^i \sim \pi_{goal}(\cdot|s^i, g)$. В нашем случае из данного распределения просэмплируем некоторый набор действий: $(a^i_1, \dots, a^i_m) \sim \pi_{goal}(\cdot|s^i, g)$. Далее, по данному набору действий и начальному состоянию можно получить набор состояний среды $(s^{i+1}_{k,1}, s^{i+1}_{k,2}, \dots, s^{i+1}_{k,n}) \sim \hat{T}(\cdot|s^i, a^i_k)$. Тогда, расстояние между выборкой и $s^{i+1}_{m,n}$ и целью g можно посчитать как расстояние Махаланобиса между

данной выборкой и целью. Чем меньше расстояние, тем больше вознаграждение за достижение цели. Фактически, мы выборочно оцениваем успешность достижения цели стратегией из текущего состояния.

- Либо же, можно попробовать считать L_2 норму между соседними признаковыми описаниями состояний. По набору норм оценить какую-то квантиль q (гиперпараметр) и по этому гиперпараметру определять достижение цели. Если $||s - g||_2 < q$, то считаем, что цель достигнута, иначе - не достигнута. И вознаграждаем агента в случае достижения. (Подробнее оценку расстояния мы опишем ниже).

По первичным результатам экспериментов, оба метода не приводили к сильным изменениям поведения агента, поэтому в дальнейшем мы использовали более простой и быстрый подход, основанный на квантилях.

4.3 Динамическое расстояние, основанное на приблизительном динамическом программировании

Есть набор работ, в которых рассматриваются расстояния, основанные на приблизительном динамическом программировании. К таким работам, например, относятся работы [2, 17]. Работа DDL [10] сравнивается с подходом, связанным с приблизительным динамическим программированием, и показывает превосходство своего подхода над классическим. Однако, у динамического расстояния есть набор недостатков, по сравнению с приблизительным динамическим программированием при условии применения моделей мира.

Во-первых, заметим, что в формуле 3.3 и на рисунке 5 длина траектории ограничена горизонтом воображения. А это значит, что и функция расстояния, обучаемая на этих траекториях согласно процедурам по ссылкам, будет ограничена длиной воображения. С одной стороны, можно было бы попробовать менять горизонт воображения адаптивно, т.е. устанавливать его пропорционально ожидаемой длине эпизода. Проблема же в таком подходе заключается в том, что модель мира неидеально предсказывает переходы между состояниями. С увеличением горизонта воображения ошибка в предсказанном, будущем, состоянии будет накапливаться, что приведёт к далёким от реальности предсказаниям, а, значит, повлияет и на стратегию, и на функцию расстояния. Т.е., бессмысленно устанавливать слишком большие горизон-

ты воображения.

Это значит, что при обучении функции расстояния с помощью решения задачи регрессии 3.3, предсказанное расстояние будет ограничено максимальной длиной горизонта, а, значит, ограничена и функция вознаграждения. В некотором смысле, такая функция вознаграждения будет локальной, и работать в некоторой окрестности цели. Чтобы решить эту проблему, достаточно начать обучать функцию расстояния не с помощью решения задачи регрессии, а с помощью приблизительного динамического программирования. Так мы сможем обучать функцию расстояния, которая не ограничена горизонтом воображения.

Общая идея заключается в том, что если имеется траектория, в которой агент достиг цели, то расстояние от любого состояния до целевого на один больше, чем расстояние от следующего состояния до целевого. Формально, новая задача обучения функции расстояния может быть записана как:

$$\mathbb{E}_{(s^k,a^k)\sim\tau}(Q(s^k,a^k|g) - (-1 + done(s^k,a^k,s^{k+1}|g)\cdot max_a Q(s^{k+1},a|g)))^2,$$

$$\text{где } done(s,a,s'|g) = \begin{cases} 1, & \text{if } s' \in U_\varepsilon(g) \\ 0, & \text{иначе} \end{cases}, \tag{4.1}$$

где траектория τ просэмплирована аналогично 3.3, ε - некоторая окрестность точки, внутри которой считается, что цель достигнута. (В нашем случае, данная окрестность определяется гиперпараметром).

Физический смысл этой процедуры заключается в том, что пока цель не достигнута, вознаграждение всегда будет равно -1. Тогда, модуль значения функции полезности будет соответствовать минимальному количеству шагов, которые необходимо совершить агенту, чтобы достичь целевого состояния (в случае, если стратегия идеальна; в случае если нет - количество шагов, которые необходимо совершить текущей стратегии, чтобы достичь целевого состояния). В дальнейшем значения

$$max_a\ Q(s,a|g)$$

могут быть использованы как значения расстояния из состояния s до состояния g.

Обучение любой функции полезности по уравнению Беллмана сопряжено с проблемами, как было показано в работах: [3, 4] значения Q-функции могут быть завышенными и нестабильными. Поэтому, решения этих проблем мы решили использовать метод, который нивелирует широкоизвестные проблемы с обучением функций

полезности в непрерывных по действиям средах: TD3 [8]. Также важно отметить следующий факт: актор алгоритма TD3 в нашем случае никак не действует в среде; все действия совершает актор-критик алгоритм, аналогичный реализованному в Dreamer [16]. В нашей реализации, актор TD3 лишь выполняет максимизацию функции полезности, в то время как критик TD3 (тоже не совпадающий с критиком Dreamer'a), обучается предсказывать расстояние для данной пары состояние-действие: (s, a). Актор-критик алгоритм [16] же обучается максимизировать сумму вознаграждений (т.е. отрицательную сумму расстояний), аналогично оригинальной имплементации PEG'a [22].

Теперь обсудим, как определяется константа ε, в окрестности которой цель считается достигнутой. Сначала, берётся батч траекторий из среды, собранный согласно процедуре Go-explore в PEG. Затем, считается L_2 норма между признаковым описанием состояния среды s двух последовательных состояний: s^t и s^{t+1}: $l = ||s^t - s^{t+1}||_2$ по всему батчу просэмплированных траекторий. Затем, мы сохраняем все эти разности в буфере D_l, хранящем последние k батчей разностей. Далее, мы считаем эмпирическую квантиль по этому буферу, и таким образом получаем значение ε (уравнение 4.2).

$$\varepsilon : P(l < \varepsilon) = q. \tag{4.2}$$

За этим эмпиричиским определением ε стоит мысль, что все признаковые описания одного и того же состояния среды, находятся близко; ближе, чем какая-то, достаточно малая, квантиль разности одношаговых переходов.

С помощью этой квантили мы определяем следующую функцию вознаграждения:

$$r(s, a, s'|g) = \begin{cases} \frac{-||s'-g||_2}{\varepsilon}, & \text{if } s' \in U_\varepsilon(g) \\ -1, & \text{иначе} \end{cases}, \tag{4.3}$$

на которой и обучаем алгоритм TD3. Это определение вознаграждения отличается от вышеописанного 4.1. Причина заключается в том, что мы хотим, чтобы расстояние, с помощью которого расчитывается вознаграждение для актор-критик алгоритма Dreamer'a [16], побуждало агента максимально точно достигать цели, а не находится в некоторой окрестности целевого состояния. Также возможны случаи, когда целенаправленная стратегия не достигнет ни одного состояния, или достигнет малого числа состояний на траектории. В этом случае, может быть полезно применение алгоритма HER [7]. Здесь идея состоит в том, что мы можем заменить целевое состояние на

состояние, достигнутое агентом. В процедуре обучения будут явно присутствовать пары состояние-цель с совпадающими состоянием, целью и положительной наградой, и, возможно, в процессе оптимизации аппроксиматору будет проще выучить, что для максимизации вознаграждения необходимо достичь целевого состояния. В дальнейшем, этот алгоритм мы будем называть DADP (Distance based on Approximate Dynamic Programming - расстояние, основанное на приблизительном динамическом программировании). В итоге, получаем алгоритм обучения расстояния, схема которого представлена ниже на рис. 10

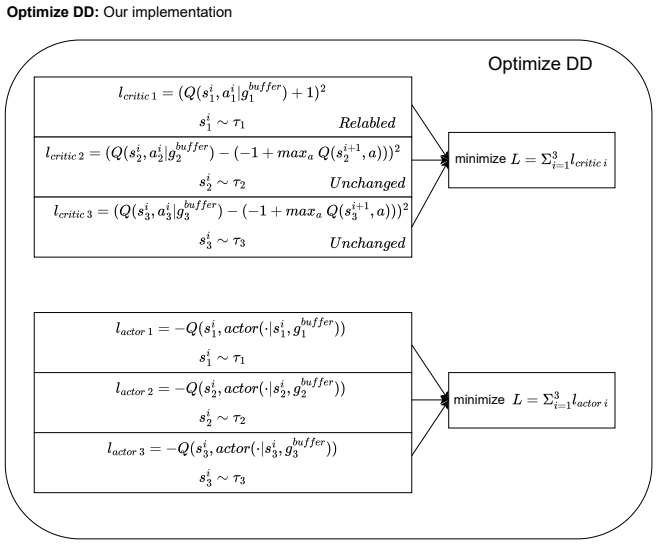

Рис. 10: Итоговая реализация блока подсчета расстояния с помощью приблизительного динамического программирование (для полной процедуры обучения см Fig. 5). В этом примере, цель первой траектории заменена на следующее состояние траектории.

Также как и в случае с динамическим расстоянием, мы рассматриваем для этого расстояния два способа обучения: на признаковом описании состояний s и наблюдений e.

5 Вычислительные эксперименты

Во время запуска экспериментов с исходной кодовой базой из PEG [22] мы столкнулись с проблемой воспроизведения точных значений метрик. Поэтому, мы попробовали подобрать гиперпараметры метода для улучшения работы метода. В этой секции представлены результаты для этой, модифицированной версии.

5.1 Точечный лабиринт

Первая серия экспериментов запущена на среде точечный лабиринт, впервые представленной в статье [13]. Наблюдения содержат в себе только текущую позицию агента $o = (x, y)$. Действия представляют собой вектор $a = (dx, dy)$, распределённый в квадрате $[-0.95; 0.95]^2$. Целью является достижение произвольного состояния в лабиринте размера 10×10. Начальное состояние эпизода всегда соответствует расположению агента в левом нижнем углу. Все эксперименты усреднялись по шести запускам. Усреднение с окном 5 применялось для всех метрик.

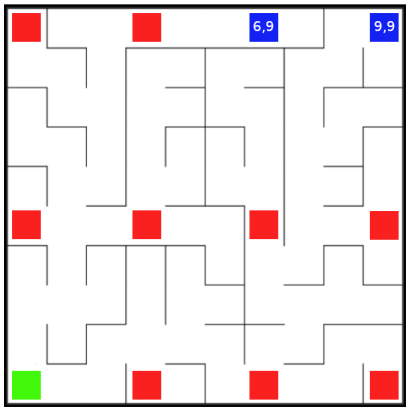

Рис. 11: Распределение целей (изображены красными и синими квадратами). Самые далёкие цели - (6, 9) и (9, 9) изображены синим. Начальное положение агента изображено зелёным квадратом.

5.1.1 Использование аналитических расстояний и преемственных представлений

Первая секция экспериментов посвящена самому простому способу подсчёта расстояний, а также попытке использования преемственных представлений. Для начала, предлагаем посмотреть карту расстояния, полученную для DD, обученного на признаковом описании наблюдений e (Рис. 12). Из данной карты расстояний видно несколько проблем, присущих оригинальной реализации функции расстояния:

1. Расстояние предсказываемое вдалеке от цели, примерно одинаково (желтый участок в нижней стороне карты).

2. Расстояние между целевым состоянием (6, 9) и состояниями ниже цели предполагается низким, хотя на самом деле это не так (состояния ниже красных,

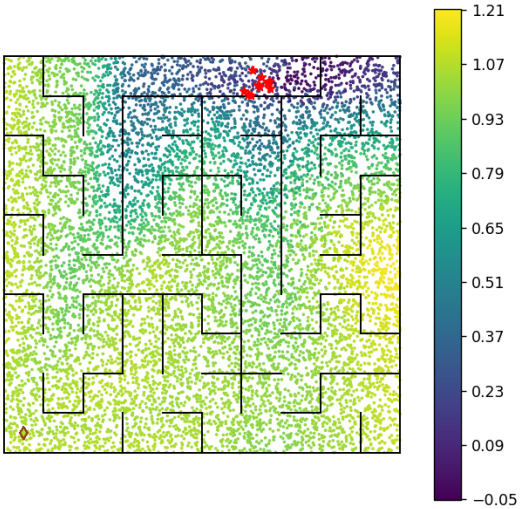

Рис. 12: Красным изображено распределение целевых состояний, цветом отмечено расстояние от точки до целевого состояния.

разделенных стеной). На самом деле, чтобы попасть из этих состояний в целевое необходимо вернутся в начальное состояние, и затем уже достичь цели по другому маршруту. То же самое можно сказать и про расстояние между состояниями (6, 9) и (9, 9), которое тоже изображено близким (фиолетовые точки).

Мы выдвинули гипотезу, что такое поведение связано с ложной экстраполяцией аппроксиматора: в случае, если негативного сэмплирования нет, динамическое расстояние может предсказывать произвольное расстояние на парах состояний, которые никогда не встречаются в одной траектории. Стоит отметить, что в алгоритме обучения динамического расстояния в реализации PEG реализован простой способ генерации негативных примеров: сэмплируется пара признаковых описаний наблюдений/состояний из разных траекторий, и полагается, что расстояние между этими состояниями равняется максимальному. Однако, как мы можем видеть из рис. 12, такой метод негативного сэмплирования всё равно не решает проблему ложной экстраполяции.

В качестве первичного решения предлагается использовать признаковое описание состояний, а не наблюдений, и на этом признаковом описании использовать различные виды расстояний между представлениями: например, косинусное расстояние / L_2 норму между представлениями. Однако использование аналитических расстояний для подсчёта расстояния до цели обладает следующей проблемой: карты рассто-

яний получаются чересчур гладкими, и, как следствие, агент не может качественно достичь целевого состояния. На Рис. 13, 14 приведен пример гладкости подсчитанных репрезентаций, и, как следствие, проблемы с четким достижением целей в среде (Рис. 15):

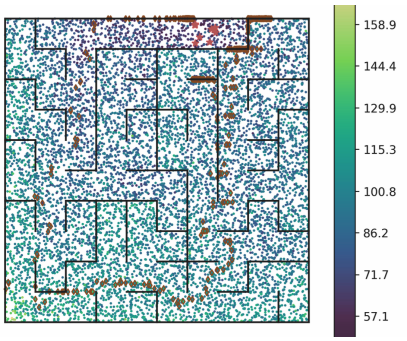

Рис. 13: Распределение расстояния до цели (6, 9). Расстояние считается как L_2 норма между текущим и целевым состоянием.

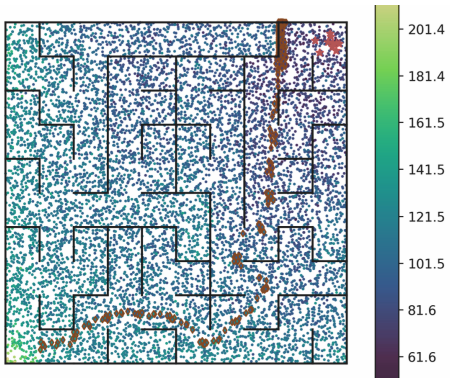

Рис. 14: Распределение расстояния до цели (9, 9). Расстояние считается как L_2 норма между текущим и целевым состоянием.

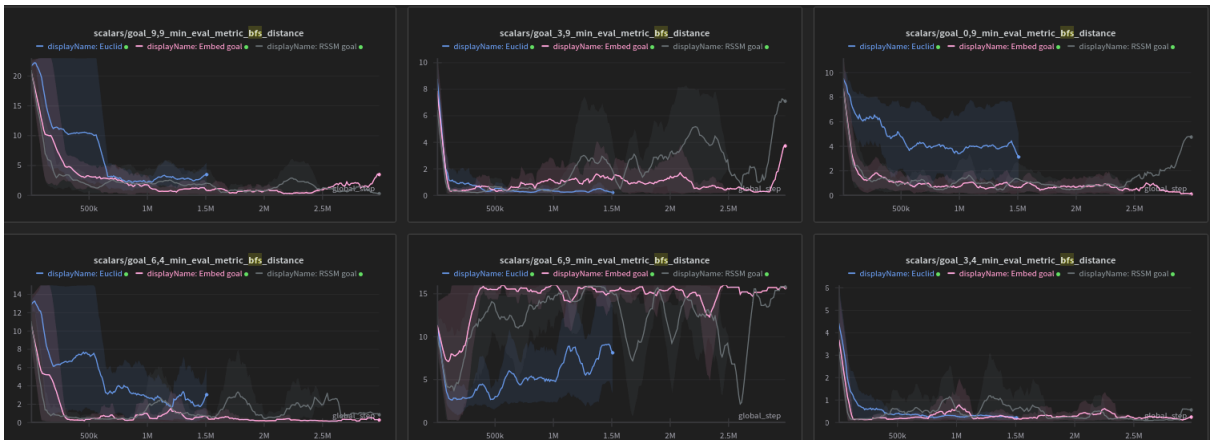

Рис. 15: Иллюстрация расстояния в среде до цели. Синим изображено расстояние при обучении агента с помощью L_2 нормы между RSSM-состояниями, розовым и серым - различные вариации динамического расстояния.

Как можно видеть, основная проблема L_2 нормы при использовании в качестве целенаправленного вознаграждения - слишком гладкая функция вознаграждения. По итогу, агент не может точно достичь целевого состояния, что приводит к завышенному расстоянию до цели с одной стороны (с.м. Рис. 15, графики для целей (0, 9), (6, 4), (6, 9)). С другой стороны, использование такого расстояния позволяет агенту,

в среднем, двигаться в направлении цели, не индуцируя точного достижения.

Иной способ избавиться от проблемы ложной экстраполяции - использование преемственных признаков. Поскольку преемственные признаки, в простом виде, хранят в себе информацию о всей траектории (дисконтированная сумма будущих признаковых описаний наблюдений), то из близости наблюдений уже не следует близость преемственных признаков: для близких наблюдений траектории не обязаны совпадать. Мы опробовали два способа реализации преемственных признаков:

1. Подсчёт функции полезности с помощью преемственных признаков (ур. 3.4) (где значения функции полезности - расстояния).

2. Подсчёт меры близости текущего состояния к целевому засчёт сравнения косинусной близости целевого состояния и усреднённого представления преемственных признаков: $\frac{SF(s_t)}{1-\gamma}$.

Но по результатам наших экспериментов, оба метода показали метрики ниже, чем у классического динамического расстояния. Причиной плохого поведения первого метода может быть отсутствие гарантий сходимости, в то время как проблемой второго метода может быть нестабильность использования преемственных признаков с моделями мира: при появлении нового прецендента представления модели мира, в теории, могут изменяться, а значит и будет изменяться преемственное представление многих состояний, что будет замедлять процесс обучения. В дальнейшем мы отказались от использования аналитических расстояний и преемственных признаков для обучения функции расстояния.

5.1.2 Анализ динамических расстояний на точечном лабиринте

В нижестоящих экспериментах представлены результаты для четырёх версий: расстоянии, обученном с помощью регрессии (DD) и с помощью динамического программирования (DADP). Оба расстояния обучались или на признаковом описании наблюдений e, или на признаковом описании состояний s. Для всех расстояний, как упоминалось выше, мы используем также локальную награду (4.3) для более адекватного достижения целей в среде.

Для начала выделим интересное наблюдение: для всех 'не самых отдалённых' целей (зеленые квадраты на рис. 11) все методы показывают стабильную сходимость, и более-менее хорошо достигают поставленной цели (Рис. 16).

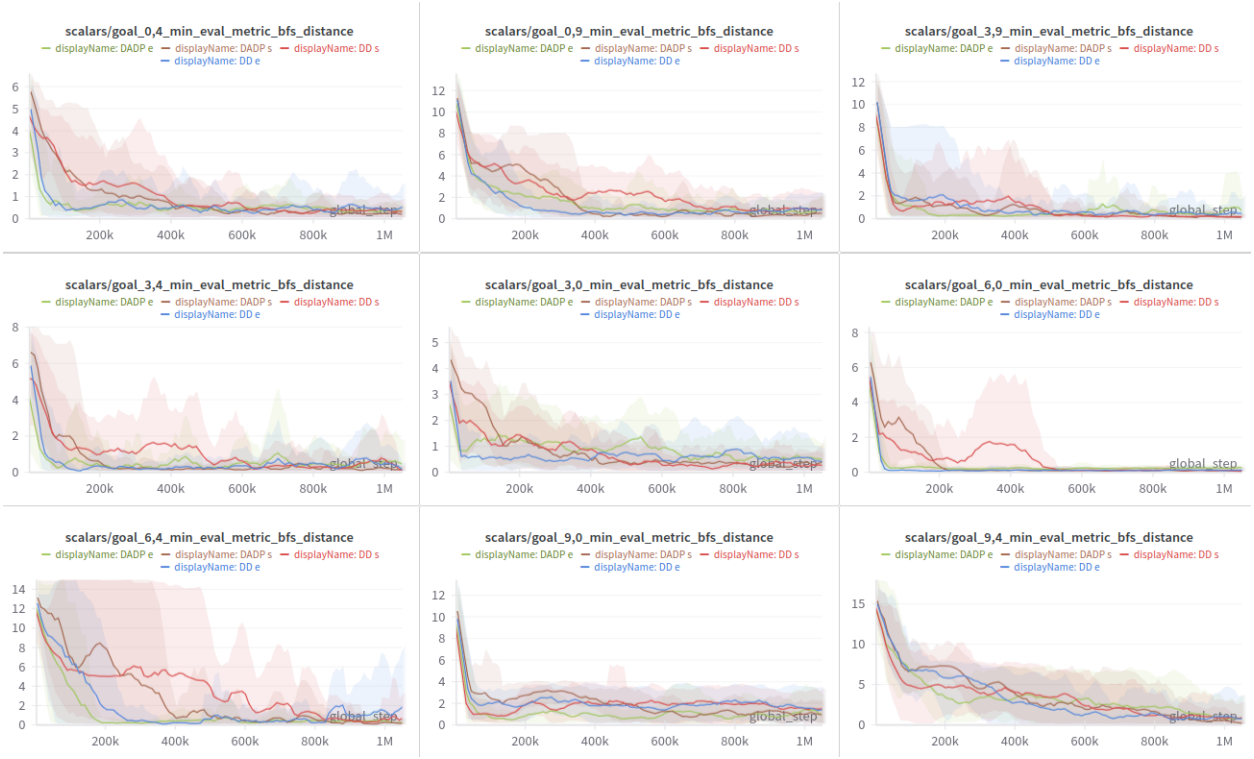

Рис. 16: Метрика расстояния (в количестве клеток между текущим положением агента и целевым) в среде до всех целей. Как можно видеть, вне зависимости от вариации (их здесь четыре), агенты, обученные на любом из расстояний, в асимптотике обучаются достигать поставленной цели.

Поэтому, в дальнейшем сконцентрируемся на паре самых сложных целей: (6, 9) и (9, 9). Сложность цели (9, 9) заключается в максимальной удалённости от начального состояния. Сложность цели (6, 9), помимо удалённости, заключается ещё и в том, что есть несколько способов приблизиться к этой цели (в смысле евклидовой близости наблюдения к цели), в то время как достичь цели можно лишь единственным способом (рис. 17).

Как можно заметить из рис. 18, идеальная карта расстояния получилась лишь для DADP s. Для расстояний, использующих в качестве входа признаковое описание наблюдений: DADP e, DD e, карты расстояний неточными: как видно, расстояние экстраполируется на состояния, разделённые стеной. Для DD s, видно что карта расстояния получилась слишком гладкой. Мы связываем такую гладкость с тем фактом, что это расстояние ограничено горизонтом воображения, из-за чего множество состояний имеет одинаковое, максимальное расстояние до цели. Логично предположить, что мы увидим идеальное поведение агента, обученного на расстоянии DADP s. Однако, на практике мы не наблюдаем идеального поведения агента. Как можно видеть

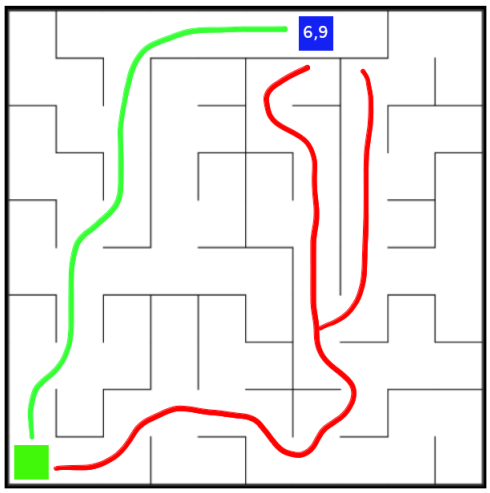

Рис. 17: Зелёным изображена траектория, ведущая к цели, красным - пара траекторий, ведущих в тупик, однако также приближающих наблюдение к целевому.

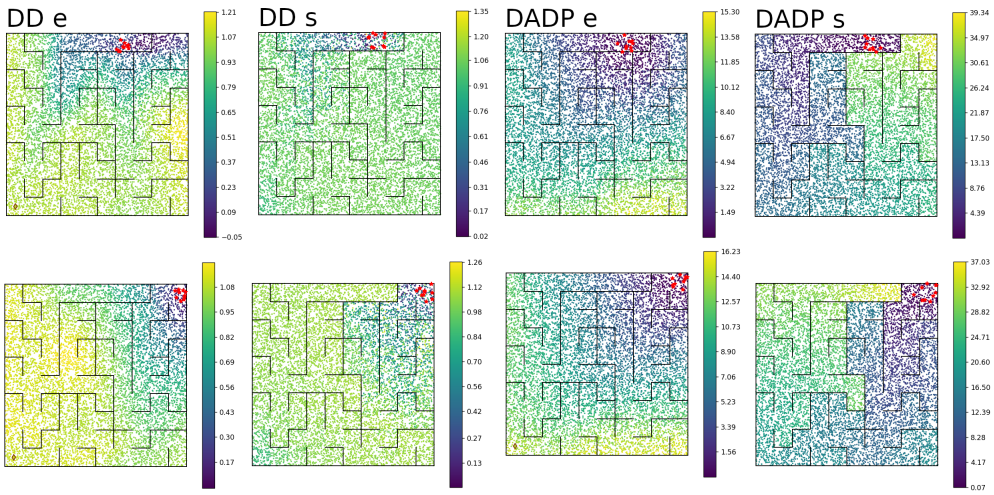

Рис. 18: Карты расстояний для каждого из методов. Красными точками изображены целевые состояния: в первом ряду цель находятся в клетке (6, 9), во втором - в клетке (9, 9). Цветом точки изображено расстояние от этой точки до одной из целевых, красных.

на рис 19, действительно, при обучении на награде, посчитанной по DADP s, агент стабильно достигает самого низкого (среди всех четырёх методов) расстояния до цели. Но с другой стороны, видно что всё равно расстояние до цели не сходится в ноль: так, для цели (6, 9) агент, в среднем, находится на расстоянии 5 до цели, а в случае цели (9, 9) находится на расстоянии 2-3 ячейки до цели. Мы предполагаем, что это может быть связано с неидеальной аппроксимацией моделью динамики среды.

Посмотрим теперь на целевую метрику: считается, что мы достигли цели в том

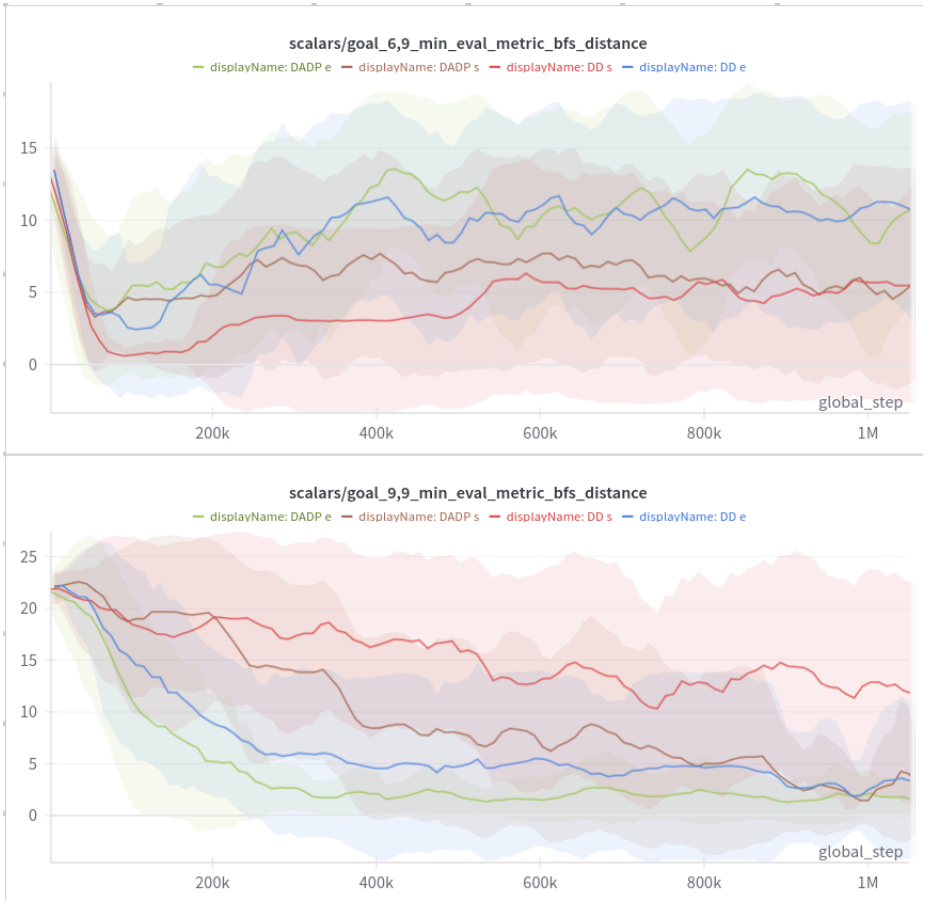

Рис. 19: Среднее расстояние до цели (измеренное в количестве клеток от состояния до целевого состояния).

случае, если агент оказался в той же ячейке, что и поставленная цель (рис. 20).

Как можно видеть из рис. 20 действительно, использование этой функции расстояния влияет и на целевую метрику в лучшую сторону. Т.е. помимо глобального достижения цели удалось добиться и адекватного локального достижения цели.

5.1.3 Анализ динамических расстояний на среде со сложным управлением

Для усложнения управления, мы выбрали в качестве управляемого объекта муравья. Как и в предыдущем разделе, задача агента - научиться ориентироваться в лабиринте. Но нельзя назвать эту среду однозначно более сложной, чем точечный лабиринт: сама конфигурация лабиринта более проста. Пространство наблюдений представляет из координату туловища агента и информацию обо всех сочленениях агента: $o \in \mathcal{R}^{29}$. Пространство действий же представляет из себя усилие, подаваемое на каждое из сочленений муравья: $a \in \mathcal{R}^{8}$. Целью является принятие какой-то позы

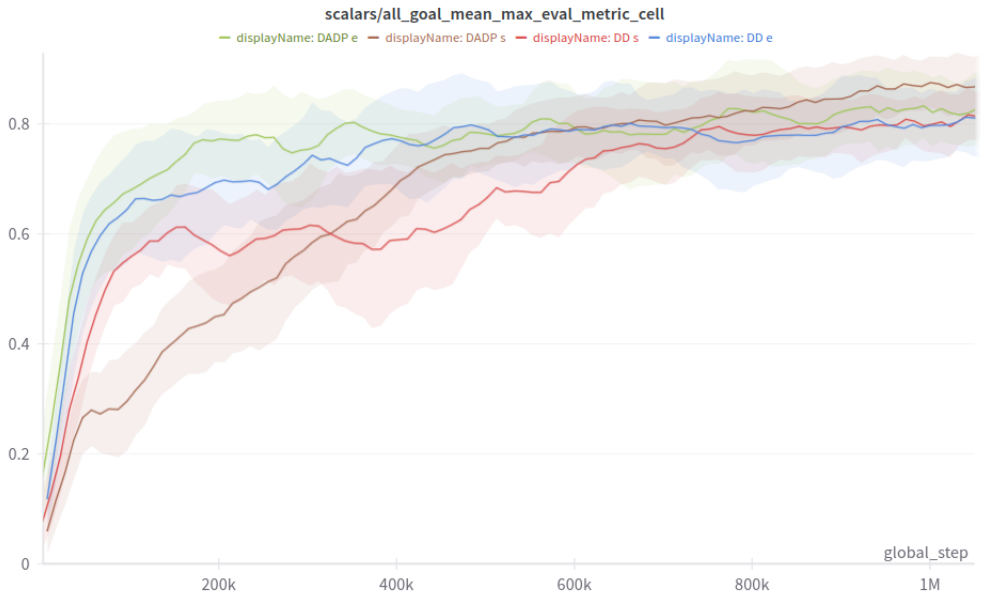

Рис. 20: Средняя метрика успеха достижения всех целей в точечном лабиринте.

в выбранной точке лабиринта (Рис. 21).

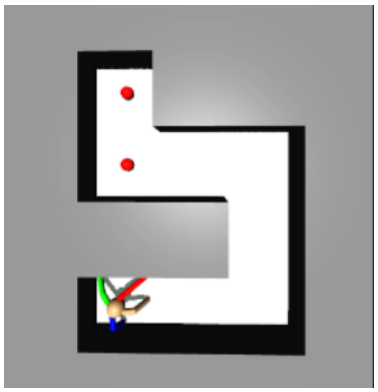

Рис. 21: Распределение целей (изображены красными точками). Начальное положение агента в самой нижней точке лабиринта. Цель - попадание в одну из красных точек и достижение одной из четырёх поз. Таким образом, количество целей в этой среде равно 8.

Как можно видеть из рис. 22, все методы обучаются приближаться к поставленной цели. Т.е., как и выше, глобальное достижение целей для каждого из методов происходит успешно.

Как можно видеть из графика 23, есть несколько отличий от графиков из точечного лабиринта. Во-первых, дисперсия графиков успешности достижения цели выше. Во-вторых, здесь подход DADP показывает себя хуже, чем DD. Мы это связываем

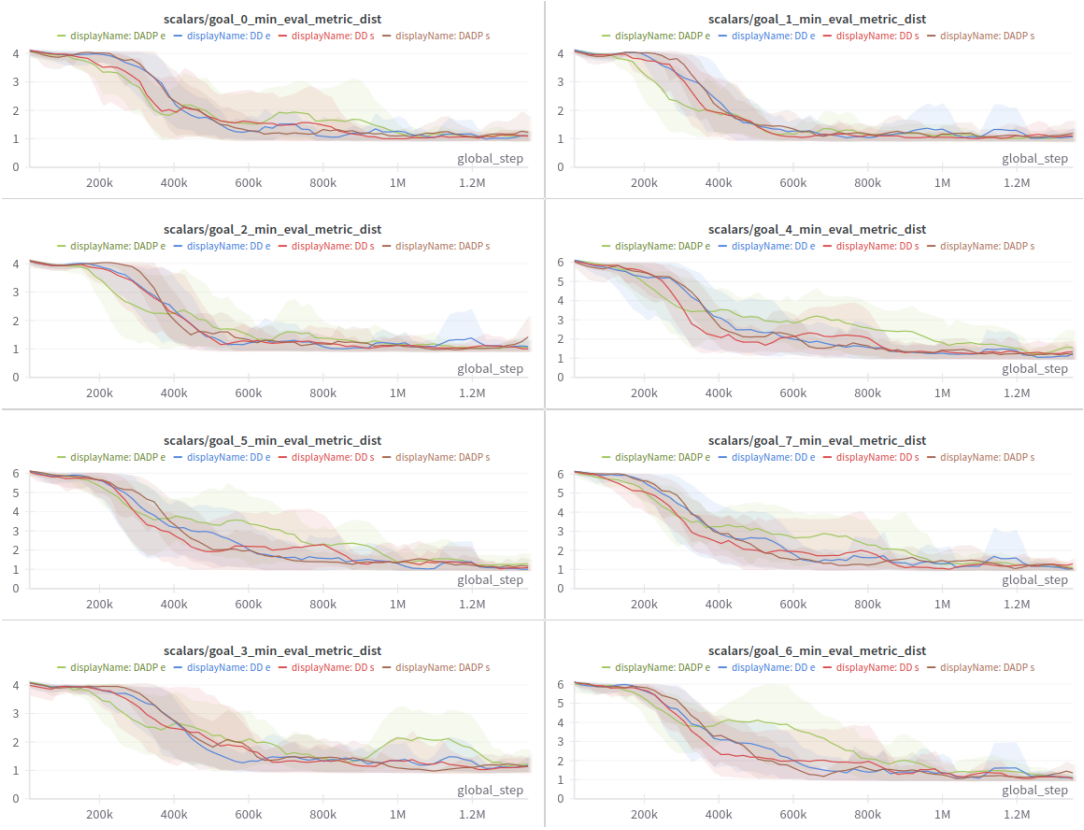

Рис. 22: Расстояние до каждой из целей.

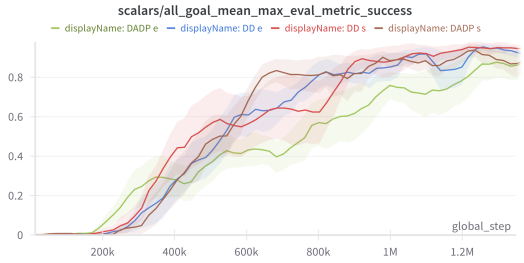

Рис. 23: Метрика успешности достижения поставленной цели и позы в лабиринтной среде с муравьев в качестве агента.

с тем фактом, что процедура обучения расстояния с помощью динамического программирования, на которой основан DADP - это более неустойчивый процесс, чем обучение расстояния с помощью решения задачи регрессии, на которой основан DD, а отклик среды на переданное управление стал более сложным. С другой стороны, в силу самой простоты среды, многие нюансы, которые делают DADP лучше DD здесь не проявляются: так, по большому счёту, задачей агента в этой среде является обучение передвижению из начальной точки лабиринта в конечную с выбором момента, в

который необходимо остановиться. С другой стороны, видно что применение s вместо e и тут показывает неухудшение по метрикам. Мы предполагаем, что на этой среде не проявляется положительных сторон использования DADP, поскольку топология лабиринта проста.

6 Заключение

В этой работе мы предлагаем два улучшения для обучения функций расстояния в моделях мира: использование признаковых описаний состояний вместо признаковых описаний наблюдений, в качестве входа в функции расстояния, и использование динамического программирования. Мы приводим теоретический анализ, который показывает, почему применение этих улучшений может породить лучшее поведение целенаправленной стратегии, обучаемой в моделях мира. Наши эксперименты показывают, что использование s вместо e как минимум не ухудшает, а в некоторых случаях и улучшает поведение функций расстояния, и, как следствие, позволяет лучше обучать целенаправленные стратегии в моделях мира.

С другой стороны, использование DADP даёт противоречивые результаты: с одной стороны, на среде точечного лабиринта использование DADP с признаковым представлением состояния в качестве входа, порождает более качественное поведение целенаправленной стратегии. Однако на среде с усложнённым управлением, использование DADP приводит к ухудшению метрик точного достижения целей. Это может быть объяснено нестабильностью обучения с помощью динамического программирования. С другой стороны, не до конца ясно, почему, при хороших картах расстояний на среде точечного лабиринта, целенаправленная стратегия не может обучиться идеальному достижению целей. Одно из направлений будущей работы может быть изучение причин вышеописанного поведения.

Как вывод, мы можем дать рекомендацию к использованию признакового описания состояния в качестве входа в функции расстояния. С другой стороны, использование DADP может быть оправдано, но результаты применения могут сильно зависеть от среды.

Работа выполнялась с использованием инфраструктуры Центра коллективного пользования «Высокопроизводительные вычисления и большие данные» (ЦКП «Информатика») ФИЦ ИУ РАН (г. Москва).

Список литературы

1. *Dayan P.* Improving Generalization for Temporal Difference Learning: The Successor Representation // Neural Computation. — 1993. — Т. 5, № 4. — С. 613—624. — DOI: 10.1162/neco.1993.5.4.613.
2. *Kaelbling L. P.* Learning to Achieve Goals // International Joint Conference on Artificial Intelligence. — 1993. — URL: https://api.semanticscholar.org/CorpusID:5538688.
3. *Hasselt H. van*, *Guez A.*, *Silver D.* Deep Reinforcement Learning with Double Q-learning // CoRR. — 2015. — Т. abs/1509.06461. — arXiv: 1509.06461. — URL: http://arxiv.org/abs/1509.06461.
4. Human-level control through deep reinforcement learning / V. Mnih [и др.] // Nature. — 2015. — Февр. — Т. 518, № 7540. — С. 529—533. — ISSN 1476-4687. — DOI: 10.1038/nature14236. — URL: https://doi.org/10.1038/nature14236.
5. *Wang Z.*, *Freitas N. de*, *Lanctot M.* Dueling Network Architectures for Deep Reinforcement Learning // CoRR. — 2015. — Т. abs/1511.06581. — arXiv: 1511.06581. — URL: http://arxiv.org/abs/1511.06581.
6. Deep Successor Reinforcement Learning / T. D. Kulkarni, A. Saeedi, S. Gautam, S. J. Gershman. — 2016. — arXiv: 1606.02396 [stat.ML].
7. Hindsight Experience Replay / M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, W. Zaremba // CoRR. — 2017. — Т. abs/1707.01495. — arXiv: 1707.01495. — URL: http://arxiv.org/abs/1707.01495.
8. *Fujimoto S.*, *Hoof H. van*, *Meger D.* Addressing Function Approximation Error in Actor-Critic Methods // CoRR. — 2018. — Т. abs/1802.09477. — arXiv: 1802.09477. — URL: http://arxiv.org/abs/1802.09477.
9. Deep Dynamics Models for Learning Dexterous Manipulation / A. Nagabandi, K. Konolige, S. Levine, V. Kumar // CoRR. — 2019. — Т. abs/1909.11652. — arXiv: 1909.11652. — URL: http://arxiv.org/abs/1909.11652.

10. Dynamical Distance Learning for Unsupervised and Semi-Supervised Skill Discovery / K. Hartikainen, X. Geng, T. Haarnoja, S. Levine // CoRR. — 2019. — T. abs/1907.08225. — arXiv: 1907.08225. — URL: http://arxiv.org/abs/1907.08225.

11. *Eysenbach B.*, *Salakhutdinov R.*, *Levine S.* Search on the Replay Buffer: Bridging Planning and Reinforcement Learning // CoRR. — 2019. — T. abs/1906.05253. — arXiv: 1906.05253. — URL: http://arxiv.org/abs/1906.05253.

12. Go-Explore: a New Approach for Hard-Exploration Problems / A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, J. Clune // CoRR. — 2019. — T. abs/1901.10995. — arXiv: 1901.10995. — URL: http://arxiv.org/abs/1901.10995.

13. Keeping Your Distance: Solving Sparse Reward Tasks Using Self-Balancing Shaped Rewards / A. Trott, S. Zheng, C. Xiong, R. Socher // CoRR. — 2019. — T. abs/1911.01417. — arXiv: 1911.01417. — URL: http://arxiv.org/abs/1911.01417.

14. Learning Latent Dynamics for Planning from Pixels / D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, J. Davidson // Proceedings of the 36th International Conference on Machine Learning. Т. 97 / под ред. K. Chaudhuri, R. Salakhutdinov. — PMLR, 09–15 Jun.2019. — С. 2555—2565. — (Proceedings of Machine Learning Research). — URL: https://proceedings.mlr.press/v97/hafner19a.html.

15. Self-supervised Learning of Distance Functions for Goal-Conditioned Reinforcement Learning / S. Venkattaramanujam, E. Crawford, T. Doan, D. Precup // CoRR. — 2019. — T. abs/1907.02998. — arXiv: 1907.02998. — URL: http://arxiv.org/abs/1907.02998.

16. Mastering Atari with Discrete World Models / D. Hafner, T. P. Lillicrap, M. Norouzi, J. Ba // CoRR. — 2020. — T. abs/2010.02193. — arXiv: 2010.02193. — URL: https://arxiv.org/abs/2010.02193.

17. *Micheli V.*, *Sinnathamby K.*, *Fleuret F.* Multi-task Reinforcement Learning with a Planning Quasi-Metric // CoRR. — 2020. — T. abs/2002.03240. — arXiv: 2002.03240. — URL: https://arxiv.org/abs/2002.03240.

18. Discovering and Achieving Goals via World Models / R. Mendonca, O. Rybkin, K. Daniilidis, D. Hafner, D. Pathak // CoRR. — 2021. — T. abs/2110.09514. — arXiv: 2110.09514. — URL: https://arxiv.org/abs/2110.09514.

19. *Pinneri C.*, *Martius G.*, *Krause A.* Neural All-Pairs Shortest Path for Reinforcement Learning // Deep Reinforcement Learning Workshop NeurIPS 2022. — 2022. — URL: https://openreview.net/forum?id=w3jZFKGLrJ.

20. Walk the Random Walk: Learning to Discover and Reach Goals Without Supervision / L. Mezghani, S. Sukhbaatar, P. Bojanowski, K. Alahari. — 2022. — arXiv: 2206.11733 [cs.LG].

21. *Gallouédec Q.*, *Dellandréa E.* Cell-Free Latent Go-Explore. — 2023. — arXiv: 2208.14928 [cs.LG].

22. Planning Goals for Exploration / E. S. Hu, R. Chang, O. Rybkin, D. Jayaraman. — 2023. — arXiv: 2303.13002 [cs.LG].

23. Distributional Distance Classifiers for Goal-Conditioned Reinforcement Learning / R. T. Akella, B. Eysenbach, J. Schneider, R. Salakhutdinov. — 2024. — URL: https://openreview.net/forum?id=qofh48zW3T.

24. Hierarchical Reinforcement Learning from Demonstration via Reachability-Based Reward Shaping / X. Gao, J. Liu, B. Wan, L. An // Neural Processing Letters. — 2024. — Май. — Т. 56, № 3. — С. 184. — ISSN 1573-773X. — DOI: 10.1007/s11063-024-11632-x. — URL: https://doi.org/10.1007/s11063-024-11632-x.

25. Mastering Diverse Domains through World Models / D. Hafner, J. Pasukonis, J. Ba, T. Lillicrap. — 2024. — arXiv: 2301.04104 [cs.AI].