

Федеральное государственное учреждение
Федеральный исследовательский центр
«Информатика и Управление»
Российской академии наук

На правах рукописи

Панов Александр Игоревич

Методы и алгоритмы нейросимвольного обучения и планирования
поведения когнитивных агентов

Специальность: 1.2.1. Искусственный интеллект и машинное обучение

ДИССЕРТАЦИЯ
на соискание учёной степени
доктора физико-математических наук

Москва — 2024

Оглавление

	Стр.
Введение	5
Глава 1. Обучение с подкреплением на основе модели среды	20
1.1 Безмодельное обучение с подкреплением	20
1.1.1 Принятие решений в условиях неопределённости	20
1.1.2 Марковский процесс принятия решений	24
1.1.3 Обучение на основе функции полезности	27
1.1.4 Параметризация и градиент стратегии	31
1.1.5 Методы «актора-критика»	35
1.1.6 Оптимизация градиента стратегии	38
1.2 Планирование поведения по известной модели	44
1.2.1 Планирование при классических допущениях	44
1.2.2 Представление состояний в задаче планирования	46
1.2.3 Планирование в условиях неопределённости	54
1.3 Интеграция планирования и обучения	59
1.3.1 Обучение модели среды	60
1.3.2 Обучение представлений для модели среды	64
1.3.3 Особенности планирования по обучаемой модели	68
1.3.4 Интеграция планирования и обучения в едином цикле	73
1.4 Выводы	77
Глава 2. Нейросимвольная архитектура для планирования и обучения	79
2.1 Когнитивные архитектуры	80
2.2 Архитектуры STRL и STRL2	83
2.2.1 Архитектура STRL для управления группой сложных технических объектов	85
2.2.2 Расширенная версия архитектуры STRL	101
2.3 NSLP: нейросимвольная архитектура для планирования и обучения	103
2.3.1 Класс задач, решаемых с использованием архитектуры NSLP	103
2.3.2 Общая схема архитектуры NSLP	106
2.3.3 Планирование и обучение в нейросимвольной архитектуре	110
2.3.4 Иерархическое обучение с использованием объектной модели среды	116
2.3.5 Принцип эквивалентности и объектная декомпозиция	121
2.4 Планирование с языковыми моделями в NSLP	126
2.4.1 Применение предобученных больших языковых моделей в задачах планирования	128
2.4.2 Оценка предобученных БЯМ в задачах планирования	133
2.5 Выводы	142

Глава 3. Обучение и использование модели в архитектуре NSLP	144
3.1 Модель среды в архитектуре «актор-критик»	144
3.1.1 «Актор-критик» с моделью среды	145
3.1.2 Поиск по дереву модели в критике	148
3.1.3 Экспериментальное исследование актор-критиков с моделью среды . .	149
3.2 Реализация модели в задачах управления	154
3.2.1 Использование нейросетевых аппроксиматоров дифференциальных уравнений в качестве модели динамики	156
3.2.2 Использование NODE в качестве модели в цикле обучения	157
3.2.3 Экспериментальное исследование модели NODE	159
3.3 Реализация модели в многоагентных задачах	162
3.3.1 Поиск по дереву с эвристиками в задаче построения пути	163
3.3.2 Многоцелевой многоагентный поиск по дереву с планированием во внутренней среде	174
3.4 Исследование среды в обучении с подкреплением на основе модели	185
3.4.1 Внутренняя мотивация в процессе обучения	188
3.4.2 Формирование сигнала внутренней мотивации	191
3.5 Выводы	197
Глава 4. Объектно-центричные представления в архитектуре NSLP	198
4.1 Распутанные объектно-центричные представления сцен	198
4.1.1 Квантованные объектно-центричные представления	200
4.1.2 Экспериментальное исследование квантованных объектно-центричных представлений	203
4.2 Слотовое объектно-центричное представление сцен	206
4.2.1 Объектно-центричные представления сцен	207
4.2.2 Модуль смеси слотов для объектного представления	211
4.2.3 Экспериментальное исследование модуля смеси слотов	214
4.3 Обучение с подкреплением с объектно-центричными моделями	222
4.3.1 Объектно-центричные подходы для построения модели мира	223
4.3.2 Графовый объектный актор-критик	226
4.3.3 Экспериментальное исследование графового объектного актора-критика	230
4.4 Языковые модели в качестве объектно-центричных моделей мира	236
4.4.1 Интеграция языковых моделей и обучения с подкреплением в виртуальных средах	237
4.4.2 Объектная декомпозиция плана действий в среде IGLU	241
4.4.3 Экспериментальное исследование объектной декомпозиции	247
4.5 Выводы	251

Глава 5. Обучение и планирование в системах управления мобильными робототехническими платформами	253
5.1 Программная реализация архитектуры NSLP для мобильного робота	253
5.1.1 Основные компоненты платформы STRL-Robotics	255
5.1.2 Реализация сенсорных блоков	257
5.1.3 Реализация блоков планирования	259
5.1.4 Модельный пример с использованием лифта	262
5.2 Визуальная навигация робототехнической платформы в помещениях	271
5.2.1 Задача генерации маневров мобильного робота	271
5.2.2 Классические навыки генерации действий	279
5.2.3 Обучаемые навыки генерации действий	281
5.2.4 Гибридная верхнеуровневая стратегия выбора навыков	284
5.2.5 Экспериментальное исследование метода SkillFusion	287
5.3 Программная реализация архитектуры NSLP для беспилотного автомобиля .	292
5.3.1 Реализация парковочного маневра	294
5.3.2 Экспериментальное исследование в составе платформы Apollo	300
5.4 Выводы	303
Глава 6. Когнитивные аспекты объектно-центричного представления	304
6.1 Знаковая картина мира как нейросимвольная модель	304
6.1.1 Семиотическая реализация нейросимвольной интеграции	304
6.1.2 Привязка к сенсомоторным данным	312
6.1.3 Сценарий в картине мира	318
6.1.4 Модельный пример	321
6.2 Когнитивные особенности планирования поведения с языковыми моделями .	324
6.2.1 Психологические подходы к мышлению и речи	325
6.2.2 Постановка психологического эксперимента	327
6.2.3 Результаты экспериментального исследования	331
6.3 Выводы	334
Заключение	335
Список рисунков	387
Список таблиц	397
Приложение А. Акты о внедрении и использовании результатов исследования	399

Введение

В промышленности, в сфере логистики и в системах автоматизации в последние годы наблюдается курс на повсеместную роботизацию¹. На сборочных конвейерах, на складах и в логистических центрах все большее применение находят робототехнические платформы, как стационарные, манипуляционные, так и мобильные платформы. Однако, класс решаемых с помощью них задач существенно ограничен заранее определенными и смоделированными сценариями с минимальной степенью неопределенности в среде и с максимальным уровнем детерминированности всех выполняемых действий. Данная ситуация во многом связана с низкой степенью автономности робототехнических платформ (агентов) и с низкой адаптивностью систем управления, реализующих синтез поведения (последовательности реакций или действий). Разработка методов и алгоритмов синтеза целенаправленного поведения агентов, взаимодействующих со средой, является одной из центральных тем в исследованиях по искусственному интеллекту, что неоднократно подчеркивалось в работах Д.А. Поспелова, Г.С. Осипова [555; 676; 679]. Существенный прогресс, достигнутый в направлении автоматического планирования и отмеченный в монографии М. Галлаба и П. Траверсо [283], позволил создать ряд эффективных символьных планировщиков [318; 519; 533], которые могут быть адаптированы для многоагентной постановки задачи [167; 670] и для планирования на основе прецедентов [159]. В большинстве случаев, в том числе и в современных планировщиках, используются символьные способы представления знаний на основе нотации логики предикатов [681].

Достигнутые результаты в области планирования, однако, оказались плохо применимы для робототехнических интеллектуальных агентов [396]. Во многом это связано с тем, что в области автоматического планирования понятие внешней среды используется очень ограниченно, только при рассмотрении вопросов выполнения построенного плана и принятия решений о модификации плана или полном перепланировании [283]. В действительности в робототехнике более существенной и важной является решение задачи привязки символов (symbol grounding problem), сформулированной еще С. Харнадом [153; 310; 485], для соотнесения используемых в классическом планировщике символов, представляющих знания агента об объектах внешней среды, и данных, поступающих с сенсоров и датчиков. Оказалось, что прямолинейная модификация используемых при планировании символов в направлении учета неопределенностей, возникающих при интерпретации сенсорных данных и подаче управляющих сигналов [344], не позволяет существенно улучшить эффективность классических планировщиков при работе в модельной или реальной среде.

Развитие теории интеллектуальных агентов в определении В.Б. Тарасова [684] привело к появлению более сложных агентных архитектур [235; 668], в которых, помимо модуля планирования поведения агента, стали выделяться модули процедурной и декларативной памяти, обработки (распознавания) сенсорных данных, обучения и коммуникации с

¹ Отчет World Robotics 2023 – <https://ifr.org/ifr-press-releases/news/world-robotics-2023-report-asia-ahead-of-europe-and-the-americas>.

другими агентами. Наибольшим успехом в этом направлении можно считать появление нескольких вариантов так называемых когнитивных архитектур С. Франклина, Д. Леирда, Б. Герцеля [168; 577; 611], в которых начали активно использоваться психологические представления о поведении человека, такие как понятия эпизодической и рабочей памяти, обучения с подкреплением. Несмотря на то что некоторые архитектуры, кроме концептуальной модели, имели и подробную реализацию в виде программных систем общего назначения [160; 371], практического применения, помимо проверки некоторых частных когнитивных моделей, они не получили. На их основе не удалось построить системы управления поведением воплощенных (*embodied*) агентов, к которым относятся и робототехнические агенты; в том числе осталась нерешенной и проблема привязки символов. Воплощение агента в данной работе определяется как некоторая виртуальная или физическая компонента, отделенная от среды, в которой действует агент, и обладающая определенным набором признаков, характеризующих данное воплощение. Например, для модели робота в симуляционной среде воплощение робота (его тело) обладает признаками размера, положения, веса, признаками, характеризующими его кинематику и динамику. Большинство когнитивных архитектур, за некоторыми исключениями [314; 547], оставались основанными на правилах и логике предикатов, что не позволяло корректно проводить интерпретацию поступающих сенсорных данных.

Развитие новых методов распознавания образов на основе эффективных биологически правдоподобных алгоритмов Д. Георга и Д. Хокингса [181; 280; 313] и технологий глубоких нейронных сетей [554] позволило рассмотреть для задачи привязки символов новые, более эффективные методы. Несмотря на то что до настоящего времени методы нейросимвольной интеграции [219; 284; 465] не могут быть распространены на практически интересные задачи по синтезу поведения интеллектуального агента, существенные успехи были достигнуты в области обучения с подкреплением, где нейросетевые методы анализа сенсорных данных позволили создать ряд аппроксиматоров, используемых при оценке успешности действий агента в среде [323; 629].

Успехи, достигнутые в последнее время в области синтеза поведения на основе глубокого обучения с подкреплением по работам Д. Сильверва, Т. Лилликрапа, Д. Хассабиса, В. Мниха [103; 294; 420], способствовали тому, что вопросу построения универсальных систем управления интеллектуальными агентами, способных действовать в широком спектре классов динамических сред, в том числе реальных физических, стало уделяться больше внимания. Однако проблема привязки символов по-прежнему остается основным барьером, препятствующим использованию конкретных технологий нейросимвольной интеграции, которые позволили бы в должной мере применять эффективные классические методы планирования и моделирования рассуждений для задач в средах, с которыми агент взаимодействует с помощью сенсоров, выдающих необработанные высокоразмерные данные. Одним из наиболее успешных подходов в этом направлении построения общих систем синтеза поведения является теория знаковой картины мира Г.С. Осипова [78; 674], в которой используются психологические представления о структуре деятельности человека и биологически правдоподобные модели приобретения знаний. В знаковой модели выделяются

несколько типов картин мира, в зависимости от структуры элемента индивидуального знания и его роли в реализации когнитивных процессов. Наиболее подходящим типом картины мира для моделирования целенаправленного поведения познающего (когнитивного) агента является рациональная картина мира [677; 686], модель которой достаточно точно аппроксимируется большими языковыми моделями, обученными на соответствующем наборе текстовых данных. Под когнитивным агентом здесь и далее будет пониматься агент, реализующий ту или иную когнитивную архитектуру, которая в свою очередь содержит в себе компоненты, моделирующие или воспроизводящие такие когнитивные функции человека, как планирование, запоминание, целеполагание, формирование новых понятий.

Теория интеллектуальных агентов нашла свое применение как напрямую в системах управления беспилотными транспортными средствами [123], так и косвенно через реализацию систем поддержки принятия решений оператора объекта управления. В данных прикладных системах реализованы некоторые важные архитектурные аспекты принятия решения и синтеза программы поведения агента: иерархичность в обработке сигналов и выработка управляющих воздействий, выбор рациональной тактики достижения цели, оперативное целеполагание.

Современные системы управления беспилотным транспортом и мобильными робототехническими платформами реализуют модульный подход к генерации автономного поведения [114]. 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. При усложнении задач, которые ставятся перед объектом управления, увеличивается количество необходимых подсистем и усложняется их внутренняя организация, межмодульное взаимодействие.

При этом 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 [290]. 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.

В настоящей работе предлагается новый подход к интеграции подсистем планирования поведения (т.е. действий как по перемещению, так и, например, по манипуляции предметами внешней среды) и обучения поведению, в котором формируется адаптивная стратегия по достижению поставленной перед агентом цели. Такая интеграция является естественной, так как обе подсистемы представляют собой различную реализацию модуля последовательного принятия решений. 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 настоящее время обучение с подкреплением без использования модели среды показывает впечатляющие результаты для многих задач управления поведением когнитивных агентов и их групп [4; 323; 51; 515; 74]. Впрочем, большинство из используемых в этой области методов требуют чрезвычайно большого количества эпизодов взаимодействия агента со средой. Эта так называемая проблема эффективности выборок является одним из ключевых препятствий на пути развития новых методов и их внедрения в практику. Среди подходов по решению этой проблемы необходимо отметить имитационное обучение (с использованием заранее подготовленных демонстраций) [5; 524] и обучение на основе модели [443; 54]. Последний перспективен также по той причине, что допускает возможность использования эффективных методов планирования и поиска по графу марковского процесса принятия решений [420; 567].

Построение модели с использованием аппроксиматоров является вычислительно затратной процедурой, и эффективные методы ее решения, не использующие заранее подготовленные и набранные агентом данные, только разрабатываются. Здесь перспективным видится подход по использованию объектного представления состояний среды [230; 258], который позволяет декомпозировать модель среды на частные объектные модели. Актуальной является разработка именно объектной формулировки задачи обучения с подкреплением на основе модели. Для этого резонно использовать недавние результаты по свойствам эквивалентности моделей по функции полезности [616], чтобы уточнить постановку оптимизационной задачи обучения.

Таким образом, **объектом исследований** диссертационной работы являются методы и модели генерации поведения когнитивных агентов в недетерминированных, частично-наблюдаемых, динамических средах с возможность задания цели поведения на естественном языке. **Предметом исследования** является разработка новых концептуальных и математических моделей, методов и численных алгоритмов генерации поведения когнитивных агентов в недетерминированной, частично-наблюдаемой, динамической среде с одновременным обучением и планированием, использующих концепцию нейросимвольной интеграции, объектно-центричного представления и формирования модели среды. **Фундаментальной научной проблемой**, на решение которой направлено диссертационное исследование, является теоретическое обобщение и развитие методов, моделей и технологий генерации поведения когнитивных агентов при работе в сложных динамических средах и в условиях задания целей поведения на естественном языке.

Целью диссертационной работы – повышение эффективности, адаптивности и степени автономности систем управления мобильными робототехническими системами общего

назначения за счет создания, внедрения и использования математического обеспечения для генерации поведения когнитивных агентов в недетерминированных, частично-наблюдаемых, динамических средах.

В соответствии с поставленной целью были сформулированы следующие **задачи**:

1. Развить теорию генерации поведения когнитивного агента: уточнить общий терминологический аппарат, выявить имеющиеся закономерности, принципы и правила организации исследуемой предметной области, систематизировать имеющиеся и разрабатываемые методы и модели.
2. Разработать комплекс математических моделей, методы и алгоритмы генерации поведения когнитивного агента в недетерминированной динамической среде.
3. Выработать общую методику построения архитектуры когнитивного агента, способного обучаться стратегии поведения и планировать поведение на основе модели среды.
4. Спроектировать новую нейросимвольную архитектуру одновременного обучения и планирования когнитивного агента с использованием обновляемой модели мира и языковых моделей для генерации концептуального объектно-центричного плана.
5. Разработать новые методы нейросимвольного анализа сцен, в том числе использующие распутанные, факторизованные и объектные представления.
6. Создать новые методы обучения с подкреплением на основе модели среды, в том числе с использованием объектно-центричного представления.
7. Разработать новые алгоритмы интеграции планирования и обучения с подкреплением, применимые для многоагентной постановки задач в динамических средах.
8. Разработать алгоритмический инструментарий на основе предложенного комплекса математических моделей и экспериментальные программные реализации основных алгоритмов.
9. Применить предложенную общую методику по построению архитектуры когнитивного агента в ряде практических задач в области многоагентного планирования пути, навигации мобильных роботов, одновременного планирования задач и перемещений робота с манипулятором и адаптивного планирования маневров беспилотного транспортного средства.

Методология и методы исследования. Методология исследования базируется на комплексном использовании и развитии следующих методов и научно-методического обеспечения:

- методы обучения с подкреплением на основе функции полезности и градиента стратегии, в том числе методы оптимизации второго порядка и итеративные алгоритмы динамического программирования;
- методы эвристического планирования и поиска с функцией полезности по дереву состояний-действий;

- методы контрастивного, самоконтролируемого и слабоконтролируемого машинного обучения с использованием нелинейных аппроксиматоров (нейронных сетей, в том числе глубоких);
- методы построения специфической и неспецифической обратной связи к мультимодальной модели динамической среды.

Научная новизна: в диссертации получены следующие новые научные результаты:

- предложена новая универсальная архитектура когнитивного агента, использующая принципы иерархичности, нейросимвольной обработки информации и генерации поведения, обучения с подкреплением и адаптации языковых моделей для задачи генерации верхнеуровневого плана, расширяющая класс задач, в которых возможно эффективное обучения стратегии агента и построение оптимального плана действий;
- разработаны новые математические модели, методы и алгоритмы одновременного планирования и обучения когнитивных агентов, действующих в сложной динамической среде, в том числе с участием других агентов, повышающие общую эффективность и обобщающую способность генерируемой агентом стратегии и уменьшающие время обучения (адаптации) данной стратегии;
- предложен новый подход к обучению объектно-центрических представлений визуальных сцен, основанный на модели смеси гауссовых распределений и реализующий нейросимвольный уровень в предлагаемой универсальной архитектуре когнитивного агента;
- создан новый класс методов обучения с подкреплением на основе объектной модели мира, превосходящий современные аналоги по метрикам качества обучения стратегии агента в специализированных объектно-центрических средах;
- усовершенствован ряд существующих моделей и методов обучения с подкреплением на основе модели мира, с моделями внутренней мотивации и с использованием эвристических планировщиков;
- разработан новый подход к интеграции методов планирования действий агента в среде с помощью больших языковых моделей и методов обучения с подкреплением для адаптации получаемого плана к конкретным текущим условиям среды.

Помимо перечисленных результатов в диссертации разработана программная реализация системы управления робототехническими платформами с использованием языковых моделей для подзадачи планирования, развернутая на различных платформах и впервые позволившая решать задачи генерации последовательности действий по языковой инструкции.

Теоретическая значимость диссертационной работы заключается в развитии теории генерации поведения когнитивного агента в недетерминированных динамических средах. В диссертационном исследовании предложена новая постановка задачи и предложены новые подходы к построению систем управления воплощенными когнитивными агентами. В работе предложены возможности развития методов, исследуемых в данной работе, как в рамках рассматриваемой постановки задачи, так и в рамках других классов задач обучения с подкреплением на основе модели среды. В диссертационном исследовании получены

теоретические результаты и научно-обоснованные решения, которые вносят значительный вклад в развитие методов и подходов к управлению когнитивными агентами в динамических средах.

Практическая значимость диссертационной работы определяется тем, что полученные теоретические результаты в области построения архитектур управления когнитивными агентами были реализованы в виде модульного программного комплекса на базе операционной системы ROS, полностью или помодульно использованы в ряде прикладных проектов и могут в дальнейшем применяться для целого ряда задач в области робототехники, беспилотного транспорта, автоматизации любых процессов, связанных с последовательным принятием решений с обратной связью от внешних условий. Ключевой практической задачей, на решение которой направлены полученные в диссертационной работе результаты, является повышение степени автономности любых автономных воплощенных систем, действующих в сложной динамической среде.

Практическое использование полученных в диссертационном исследовании результатов представлено следующими научно-исследовательскими работами с индустриальными заказчиками:

1. Разработка математического обеспечения для решения задач автоматического планирования маршрута и траектории движения, функционирующего в составе комплекта аппаратуры управления (КАУ) транспортного средства (компания НПК БИС, 2020 г.);
2. Исследование и разработка математического обеспечения для решения задач автоматического планирования маршрутов и траекторий движения, функционирующего в составе комплекта аппаратуры управления (КАУ) транспортного средства (компания НПК БИС, 2021–2022 гг.);
3. Разработка математического обеспечения для решения задач построения динамической карты проходимости и планирования на ее основе движения мобильных наземных роботов (ООО «Фаст Сенс Студия», 2021–2022 гг.);
4. Разработка математического обеспечения для решения задач автоматического управления движением автомобильного транспортного средства в условиях дорог общего пользования (ООО «ИнтеграНТ», 2022 г.);
5. Разработка системы управления роботом с модулем планирования по языковым инструкциям для сортировки объектов в помещении с использованием мобильного робота и манипулятора (ПАО «Сбербанк России», 2023 г.).

Основные положения, выносимые на защиту:

1. Предложена нейросимвольная архитектура управления поведением когнитивного агента, включающая в себя компоненты одновременного планирования и обучения, а также компонент концептуального планирования с использованием языковых моделей.
2. Созданы модели и методы интеграции планирования и обучения с подкреплением, в том числе с использованием модели среды, для решения сложных визуальных и

векторных задач управления поведением когнитивным агентом, включая задачи в многоагентной постановке.

3. Разработаны модели и методы объектно-центричного подхода к представлению сенсорной информации о статических сценах для использования в нейросимвольной архитектуре управления поведением когнитивного агента.
4. Созданы модели и методы объектно-центричного обучения с подкреплением с использованием динамической модели среды для интеграции планирования и обучения в нейросимвольной архитектуре управления поведением когнитивного агента.
5. Разработан программно-алгоритмический инструментарий, основанный, в том числе, на полученных теоретических результатах, для решения задачи генерации действий робототехнической платформой в сложной динамической среде, позволяющий использовать как обучаемые компоненты, так и классические планировочные. Создана экспериментальная программируемая реализация элементов данного инструментария, использующаяся для решения практических задач управления поведением.
6. Предложено использование разработанных моделей и методов одновременного планирования и обучения в ряде практически важных робототехнических задачах: навигация мобильной платформы внутри помещений, адаптивное планирование маневров беспилотным транспортным средством, перемещение и манипуляция объектами мобильной платформы по языковым инструкциям.

Ряд предложенных методов и подходов, а также их приложения были использованы при разработке учебных курсов «Машинное обучение с подкреплением», «Введение в методы искусственного интеллекта», «Интеллектуальная робототехника», «Методы искусственного интеллекта в анализе данных», «Программные средства для задач искусственного интеллекта», «Эвристические методы планирования», читаемых в рамках магистерской программы «Методы и технологии искусственного интеллекта» в МФТИ и ряда курсов, читаемых представителям промышленных компаний (ПАО «Сбербанк России»).

Положения, выносимые на защиту в диссертационной работе, соответствует паспорту специальности 1.2.1. «Искусственный интеллект и машинное обучение», в частности, пунктам:

- положение 1 соответствует пункту 3 «Методы и алгоритмы моделирования мыслительных процессов: рассуждений, аргументации, распознавания и классификации, формирования понятий»;
- положения 1 и 3 соответствует пункту 4 «Разработка методов, алгоритмов и создание систем искусственного интеллекта и машинного обучения для обработки и анализа текстов на естественном языке, для изображений, речи, биомедицины и других специальных видов данных»;
- положения 2 и 4 соответствуют пункту 5 «Методы и технологии поиска, приобретения и использования знаний и закономерностей, в том числе — эмпирических, в системах

- искусственного интеллекта. Исследования в области совместного применения методов машинного обучения и классического математического моделирования»;
- положения 1, 5 и 6 соответствуют пункту 6 «Формализация и постановка задач управления и (поддержки) принятия решений на основе систем искусственного интеллекта и машинного обучения. Разработка систем управления с использованием систем искусственного интеллекта и методов машинного обучения в том числе — управления роботами, автомобилями, БПЛА и т.п.»;
 - положения 1, 2 и 4 соответствуют пункту 7 «Разработка специализированного математического, алгоритмического и программного обеспечения систем искусственного интеллекта и машинного обучения. Методы и средства взаимодействия систем искусственного интеллекта с другими системами и человеком-оператором»;
 - положение 2 соответствует пункту 8 «Многоагентные системы и распределенный ИИ»;
 - положение 1 соответствует пункту 11 «Исследования в области “сильного ИИ”, включая формирование понятийной базы и элементов математического формализма, необходимых для построения алгоритмического аппарата».

Достоверность полученных результатов обеспечивается общей методикой проведения математического анализа разработанных моделей и методов, а также методикой численного эксперимента для конкретных элементов программно-алгоритмического инструментария. Обоснованность научных результатов и выводов, представленных в работе, определяется корректным применением апробированных нейросетевых методов, методов планирования поведения и обучения с подкреплением. Результаты находятся в соответствии с результатами, полученными другими авторами. Для каждого из элементов программно-алгоритмического инструментария предлагается его детальное описание, а также полный список гиперпараметров, используемых при обучении. Основные результаты представлены в публикациях с высоким уровнем цитируемости, также на ведущих конференциях по тематике диссертации. Программные комплексы, созданные на основе результатов, полученных в диссертационном исследовании, успешно внедрены в целом ряде организаций.

Основные результаты получены автором в рамках научной деятельности и научных проектов, поддержанных грантами Российского научного фонда (№18-71-00143 «Иерархическое обучение с подкреплением в задаче приобретения концептуальных процедурных знаний когнитивными агентами», №20-71-10116 «Обучение с подкреплением с использованием сетевых векторно-символьных представлений в задаче интеллектуальной навигации когнитивных агентов»), Российского фонда фундаментальных исследований (№18-29-22027 «Персональные когнитивные ассистенты, сопровождающие деятельность человека в информационном пространстве», №17-29-07051 «Сетевая модель знаковой картины мира и реализация в ней когнитивных функций»), Министерства науки и высшего образования Российской Федерации (проект №075-15-2024-544 «Математические модели и численные методы как основа для разработки робототехнических комплексов, новых

материалов и интеллектуальных технологий конструирования»). Результаты **реализованы и внедрены** в ряде индустриальных компаний, в таких как НПК БИС, ООО «ИнтеграНТ», ПАО «Сбербанк»(акты о внедрении и использовании результатов диссертационного исследования представлены в приложении А).

Апробация работы. Основные результаты диссертации докладывались и обсуждались на следующих международных и российских конференциях:

- международная конференция Artificial General Intelligence (AGI) 2020 [57], 2023 [66];
- международная научно-практической конференция «Интегрированные модели и мягкие вычисления в искусственном интеллекте» (2022) [96];
- национальная конференция по искусственному интеллекту с международным участием КИИ 2016 [88], 2020 [94], 2023 [97];
- международная конференция Biologically Inspired Cognitive Architectures [64];
- международная конференция Interactive Collaborative Robotics (ICR 2023) [28];
- международная конференция Hybrid Artificial Intelligent Systems (HAIS 2023) [12; 24];
- международная конференция по искусственному интеллекту (AAAI 2024) [2; 10];
- международная конференция по робототехнике и автоматизации (ICRA) [14];
- международная конференция по обучению представлений (ICLR) 2022 на воркшопе Elements of Reasoning: Objects, Structure and Causality [55], 2023 на воркшопе NeSy-GeMs [17], 2024 [16];
- международная конференция Pattern Recognition and Machine Intelligence (PReMI) 2023 [72];
- международная конференция Joint Conference on Artificial Intelligence (IJCAI) 2023 на воркшопе Neuro-Symbolic Agents [58];
- международная конференция Neural Information Processing Systems (NeurIPS) 2022 на соревновательном треке [8];
- международная конференция Mexican International Conference on Artificial Intelligence (MICAI) 2021 [73];
- международная конференция SGAI International Conference on Artificial Intelligence (SGAI) 2021[30];
- международная научная конференция Intelligent Information Technologies for Industry (IITI) 2021 [29], 2022 [27].

Также ряд результатов работы обсуждались на следующих научных мероприятиях:

- конференция разработчиков ROS Meetup (февраль 2023, февраль и апрель 2024);
- выставка Россия на ВДНХ (февраль 2024);
- международная конференция AI Journey (ноябрь 2021, ноябрь 2022, ноябрь 2023);
- конференция Kaspersky Neuromorphic AI (ноябрь 2023);
- конференция Fall Into ML в ВШЭ (октябрь 2023);
- международная конференция Нейроинформатика (октябрь 2022, октябрь 2023);
- летняя школа молодых ученых AIRI Summer (июнь 2022, июль 2023)
- летняя школа молодых ученых РАИИ (июль 2021, июль 2022, июль 2023);
- молодежный международный экономический форум (июнь 2023);

- выездная секция AIJ во Владивостоке (июнь 2023);
- научный семинар AIRI Иишница (июнь 2023);
- международная конференция OpenTalks.AI (февраль 2021, февраль 2022, март 2023);
- семинар «Ключевые опережающие научные инициативы» в МИФИ (апрель 2022, январь 2023);
- выездная секция AIJ в Новосибирске, НГТУ (декабрь 2022);
- научный форум Наука 0+ (октябрь 2022);
- семинар цикла AGI-in-Russian (январь 2022);
- научная конференция Vladivostok AI Week (ноябрь 2021);
- балтийский форум: нейронаука, искусственный интеллект и сложные системы BF-NAICS (сентябрь 2021);
- образовательный форум олимпиад Я-профессионал «Математика и искусственный интеллект» (апрель 2021);
- цикл вебинаров VSAONLINE (март 2021);
- российско-французский Форум по искусственноому интеллекту (февраль 2021);
- общемосковский научный семинар «Проблемы искусственного интеллекта» (январь 2021).

Часть результатов, вошедших в диссертационное исследование, были отмечены медалью Российской академии наук для молодых ученых за 2017 год, призами за лучшее решение задачи на соревнованиях NeurIPS MineRL 2019 (команда CDS) и CVPR Habitat 2023 (команда SkillFusion).

Публикации. Материалы диссертации опубликованы в **98** рецензируемых печатных работах, относящихся к следующим категориям:

- **54** статьи в изданиях из собственного перечня журналов МФТИ категории К1: [1; 3–7; 9; 12; 13; 15; 18; 20; 22; 24; 28; 30–33; 35–37; 39; 42–46; 49; 51; 52; 54; 57; 60–62; 66–77; 82; 84; 87; 89; 95; 98];
- **8** статей в изданиях, приравненных к журналам перечня ВАК категории К1: [21; 23; 26; 34; 38; 47; 48; 59];
- **7** статей в изданиях, приравненных к журналам перечня ВАК категории К3: [11; 25; 27; 29; 40; 50; 56];
- **4** статьи в трудах конференций уровня А*, индексируемых в Web of Science и/или Scopus: [2; 10; 14; 16];
- **7** статей в остальных изданиях, индексируемых в Web of Science и/или Scopus: [8; 19; 41; 53; 63–65];
- **1** монография: [78];
- **17** статей в остальных рецензируемых изданиях: [17; 55; 58; 79–81; 83; 85; 86; 88; 90–94; 96; 97].

Личный вклад. Содержание диссертации и основные положения, выносимые на защиту, отражают персональный вклад автора в опубликованные работы. Все представленные в диссертации результаты получены лично автором. В работах [1; 2; 4–7; 9–12; 14–16; 19; 21–30; 37–41; 46–54; 56; 57; 62; 66; 67; 69–72; 74–77; 87; 92] автором были

проведены постановка задач, определение методов решения, анализ результатов. В работах [13; 20; 31–36; 42–45; 59–61; 63–65; 68; 78–86; 88–91; 93; 95; 96; 98] автор разработал математические методы и алгоритмы с программной реализацией отдельных компонентов. В работе [91] автор провел анализ существующих когнитивных архитектур. В работах [13; 64; 88; 89; 95; 98] автором предложена архитектура управления STRL и формализован ее стратегический уровень. В работах [61; 96] автором предложена архитектура NSLP, доказаны формальные свойства ее работы в режиме обучения. В работах [62; 66] автором предложена концепция высокоуровневого планирования поведения с использование больших языковых моделей. В работах [18; 57; 94] автором предложена и описана эффективная реализация архитектуры актора-критика с использованием модели среды. В работах [22; 28] автор предложил концепцию интеграции планирования по модели и обучения с подкреплением с оптимизацией стратегии. В работах [2; 7; 10; 12; 67; 70; 77] автором предложена архитектура одновременного обучения и планирования в многоагентное постановке. В работах [87; 97] автором поставлена задача разработки методов внутренней мотивации для эффективного исследования среды с одновременным ее моделированием. В работе [16] автором предложена концепция нейросимвольной интеграции со слотовым представлением сцены. В работах [17; 72] автором формализована задача обучения распутанных представлений в целях нейросимвольной интеграции. В работах [55; 58] автором предложен метод использования объектных представлений в обучении с подкреплением на основе модели. В работе [8] автором предложен метод использования предобученных умений совместно с языковыми моделями. В работах [9; 49; 73; 75] автором предложена реализация архитектуры NSLP в задачах навигации. В работах [3; 76] автором предложена концепция дообучения моделей для повышения эффективности планирования. В работах [27; 29; 30] автором предложена концепция одновременного обучения и планирования в задачах управления беспилотным транспортом. В работах [24; 45; 69; 78] автором предложения семиотическая модель для реализации нейросимвольной интеграции.

Краткое изложение содержания диссертационной работы. Во введении обосновывается актуальность темы диссертации, формулируются объект исследования, цели и задачи, определяются методы исследования и описываются основные результаты, выносимые на защиту.

В главе 1 дается краткое изложение основ обучения с подкреплением как без использования модели среды, так и на основе обновляемой модели, а также методов планирования по известной модели.

Раздел 1.1 посвящен безмодельному обучению с подкреплением и его месте в классе методов, относящихся к категории подходов к принятию решений в условиях неопределенности. Рассматривается классический марковский процесс принятия решений, описывается обучение на основе функции полезности и параметризация стратегии. Даются основные теоретические основы работы архитектуры агента семейства «актор-критик», описываются современные методы оптимизации градиента стратегии.

Раздел 1.2 посвящен планированию поведения по известной модели. Даются постановки задачи и основные базовые решения для случая классических допущений и в условиях

неопределенности. Отдельно обсуждается задача представления состояний в задаче планирования поведения.

Раздел 1.3 посвящен методам и подходам к интеграции планирования и обучения в единой архитектуре агента. Обсуждаются методы обучения модели мира и отдельно задача обучения эффективных представлений для моделирования среды. Описываются особенности планирования по обучаемой модели. Данна классификация подходов к интеграции планирования и обучения в полном цикле.

В главе 2 предлагается концептуальное описание разработанной автором нейросимвольной архитектуры для планирования и обучения NSLP. Уточняется постановка задачи, для которой используется данная архитектура, проводится теоретический анализ особенностей ее работы в процессе приобретения знаний и их использования при планировании. Основные результаты главы опубликованы в работах [13; 61; 62; 64; 66; 88; 89; 91; 95; 96; 98].

Раздел 2.1 посвящен краткому обзору когнитивных архитектур, особенностей их построения и ключевых недостатков, которые привели к пересмотру подходов по построению архитектур управления поведением когнитивных агентов и необходимости развития новых нейросимвольных подходов.

Раздел 2.2 посвящен описанию предложенной автором базовой иерархической архитектуры STRL, изначально предназначеннной для управления группой сложных технических объектов. Даётся описание ее расширенной версии с обучаемыми компонентами, в том числе с использованием предобученных моделей.

Раздел 2.3 содержит основную информацию и описание NSLP архитектуры, являющейся одним из основных результатов диссертационного исследования. Описаны ее основные компоненты, приводится модельный пример, и обсуждаются теоремы о градиентах, которые формализуют работу данной архитектуры. Отдельно рассматривается вопрос об объектной декомпозиции в NSLP при планировании и обучении в контексте соответствия принципу эквивалентности модели среды.

Раздел 2.4 посвящен вопросу использования языковых моделей для верхнеуровневого планирования в NSLP. Обсуждается применение предобученных больших языковых моделей, их возможностей при генерации плана поведения и необходимости организации обратной связи.

В главе 3 рассматриваются особенности реализации процессов обучения и использования модели в архитектуре NSLP. Основные результаты главы опубликованы в работах [2; 7; 10; 12; 18; 22; 28; 57; 67; 70; 77; 87; 94; 97].

Раздел 3.1 посвящен особенностям интеграции модели среды в архитектуру семейства «актор-критик» при реализации процесса обучения архитектуры NSLP. Представлены экспериментальные результаты различных вариантов интеграции на модельных средах с обучением по наблюдениям, представленными изображениями.

Раздел 3.2 посвящен особенностям реализации архитектуры NSLP в задачах робототехнического управления. Рассматривается возможность генерации аналитического

выражения, описывающего динамику собственно робота и использования ее в качестве модели мира в стандартном цикле «обучение-планирование».

Раздел 3.3 посвящен реализации для обучения и планирования в многоагентных средах. Рассматривается подход поиска по дереву с эвристическими функциями для сокращения перебора и метод поиска по дереву Монте-Карло с маскированием действий. Представлены экспериментальные результаты на классической задаче многоагентного поиска пути в среде POGEMA.

Раздел 3.4 посвящен вопросу реализации внутренней мотивации в архитектуре NSLP при исследовании среды и эффективном обучении в ней. Предложена реализация конкретных компонентов генерации внутреннего вознаграждения и его использования при обновлении модели среды и стратегии агента.

В главе 4 обсуждается реализация нейросимвольного механизма в архитектуре NSLP с помощью объектно-центрических представлений как в слотовом, так и в факторизованном вариантах. Основные результаты главы опубликованы в работах [8; 16; 17; 55; 58; 72].

В разделе 4.1 посвящен описанию метода конструирования распутанных представлений на основе поддержания факторизации латентного пространства нейросетевого кодировщика. Описывается модель с использованием гиперразмерных векторов обладающих свойствами регуляризации представлений.

Раздел 4.2 освещается другой подход по формирования объектно-центрических представлений – метод обучения слотовых нейросетевых представлений объектных статических сцен. Описывается алгоритм с использованием модели смеси распределений. Представлено экспериментальное исследование на таких наборах данных как CLEVER, Emoji.

Раздел 4.3 посвящен использованию факторизованных моделей мира для задачи обучения с подкреплением в контексте архитектуры NSLP. На основе базового алгоритма обучения «актор-критик» предлагается использовать графовую факторизацию модели мира для разделения причинно-следственных связей, влияющих на стратегию агента.

В разделе 4.4 рассматривается использование языковых моделей в качестве планировщиков при объектно-центрической постановке задачи в таких средах как IGLU и Crafter. Обсуждаются особенности их интеграции в архитектуру NSLP с необходимой декомпозицией задачи.

В главе 5 описывается нескольких прикладных задач, для решения которых был использован алгоритмический инструментарий, созданный при реализации моделей и методов, включенных в архитектуру NSLP. Основные результаты главы опубликованы в работах [3; 9; 27; 29; 30; 49; 73; 75; 76].

Раздел 5.1 фокусируется на реализации ряда компонентов архитектуры NSLP в программной библиотеке STRL-Robotics, основанной на робототехнической операционной системе ROS. Описываются основные компоненты, реализация сенсорных компонент и компонентов управления движением по траектории. Приводится пример с использованием лифта мобильной платформой с манипулятором.

Раздел 5.2 посвящен реализации архитектуры NSLP в задаче визуальной навигации робототехнической платформы внутри помещений. Сначала ставится собственно задача генерации маневров мобильного робота для достижения конкретных объектов, описываются существующие подходы и рассматривается реализация верхнеуровневой стратегии NSLP на базе метода интеграции умений, сочетающего классические подходы по планированию перемещения и обучаемые стратегии.

Раздел 5.3 посвящен реализации элементов архитектуры NSLP в задаче генерации маневров беспилотного автомобиля на основе программной платформы Apollo. Представлен пример реализации подхода в задаче генерации маневра парковки.

В главе 6 описываются когнитивные аспекты реализации архитектуры NSLP и, в частности, семиотического механизма реализации нейросимвольной интеграции. Основные результаты главы опубликованы в работах [24; 45; 69; 78].

Раздел 6.1 посвящен теории знаковой картины мира, реализующей семиотический механизм нейросимвольной интеграции. Рассмотрена архитектура семиотического агента как варианта нейросимвольной архитектуры NSLP, проанализирована проблема привязки символов к сенсомоторным данным, представлены сценарии поведения в картине мира и дан модельный пример работы семиотического агента.

Наконец, в разделе 6.2 исследуются когнитивные особенности реализации алгоритма планирования с использованием языковых моделей. В начале раздела дано краткое описание психологических подходов к моделированию мышления и речи, а затем описывается психологический эксперимент, на основе которого возможно проанализировать особенности процесса генерации плана языковой моделью.

В заключении кратко описаны проведенные исследования и полученные результаты, а также представлен анализ дальнейших направлений работы по данной тематике.

Объем и структура работы. Диссертация состоит из введения, 6 глав, заключения. Полный объём диссертации составляет 404 страницы, включая 104 рисунка и 23 таблицы. Список литературы содержит 595 наименования.

Глава 1. Обучение с подкреплением на основе модели среды

Данная глава посвящена общим вопросам поиска оптимальной стратегии последовательного принятия решений в условиях неопределенности. Вводятся способы постановки задач планирования и обучения с подкреплением как частные подходы к решению более общей задачи принятия решений интеллектуальным агентом, действующим в среде. Рассматривается марковский процесс принятия решений как универсальный формализм, описывающий взаимодействие агента и среды. Также вводятся базовые безмодельные алгоритмы обучения на основе функции полезности, градиента стратегии и архитектуры «актор-критик». В заключение главы обсуждаются подходы к построению и обновлению модели среды в процессе обучения агентом.

1.1 Безмодельное обучение с подкреплением

1.1.1 Принятие решений в условиях неопределённости

Многие важные прикладные задачи, такие как автоматическая стыковка в космическом пространстве, управление движением транспортного средства в городских условиях, постановка медицинского диагноза, торги на фондовом рынке и поддержание работоспособности атомного реактора, предполагают *принятие решений* в условиях неопределенности. В данной работе рассматриваются методы и алгоритмы, относящиеся к классу методов автоматического принятия решений, для которых необходимо учитывать различные источники неопределенности [358]. Разнообразие источников неопределенности и различные варианты задания целевой функции при последовательном принятии решений требуют разработки новой методологии, позволяющей интегрировать различные подходы, и нового набора более эффективных алгоритмов.

В настоящей работе рассматривается конкретный вариант задачи принятия решения, в которой *агент*, принимающий решения, а именно выполняющий *действия*, в явном виде отделен от *среды*, которая реализует эти решения, т.е. меняется в результате выполнения действий агента. Агент может представлять собой физическую сущность, например, человека или мобильного робота, действующую в реальном мире, либо нефизическую, виртуальную сущность, т.е. некоторую программу, взаимодействующую с другой программой (виртуальной средой). Взаимодействие агента со средой в классических системах принятия решений реализовано в виде итеративного цикла наблюдение–действие (см. рисунок 1.1). В данном случае под действием подразумевается инициируемое агентом изменение все

системы агент-среда, т.е. действие может приводить как к изменениям (свойств) в самой среде, так и к изменениям (свойств) агента.

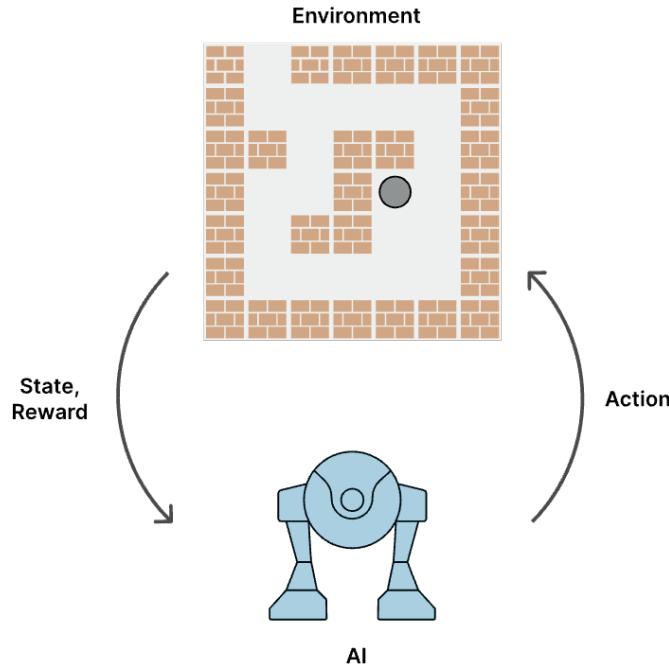


Рисунок 1.1 — Итеративный цикл взаимодействия агента (AI) и среды (Environment), включающий в себя получение информации из среды (State, Reward) и выполнение действия в ней (Action).

Формально цикл взаимодействия будет описываться следующим образом. Вводится дискретное множество моментов времени $T = \{0, 1, \dots, t, t+1, \dots\}$. В каждый момент времени t агент получает наблюдение o_t из некоторого пространства наблюдений O , которое может представлять собой, например, совокупность активных элементов сетчатки глаза человека или выходы сенсоров беспилотного автомобиля (лазерного дальномера или RGB-D камеры). В общем случае наблюдения подвержены шумам и не содержат полной информации, достаточной для принятия решения в данный момент. Например, изображение, полученное с RGB-D камеры, не содержит информации об объектах, расположенных в среде за датчиком и влияющих на агента. Агент, в свою очередь, выбирает действие a_t из заданного пространства действий A в результате некоторого *процесса принятия решений*. В общем случае действие, например, выполнение элементарного движения в пространстве, может недетерминированно изменить среду, например, положение агента.

В рассматриваемой постановке задачи подразумеваются *интеллектуальные агенты*, реализующие рациональное целенаправленное поведение в среде [675], т.е. стремящиеся достичь определенную цель за конечное время. Иными словами, при данной истории взаимодействия, т.е. последовательности пар наблюдение–действие $\{(o_t, a_t)\}$, и, возможно, некоторой дополнительной информации о среде агент должен выбрать действие a_t , оптимальное с точки зрения некоторого *критерия качества* достижения выбранной цели. Получающееся отображение пространства наблюдений в пространство действий $\pi : O \rightarrow A$ называется *стратегией принятия решений*.

Существуют различные подходы к построению архитектуры интеллектуального агента, принимающего решения в некоторой среде [684]. Ключевую роль в выборе подходов играет степень автоматизации принятия решений в сложной динамической среде и возможность работать в условиях меньшей или большей степени неопределенности. В классической теории принятия решений используются следующие основные подходы.

Прямое программирование Рассмотрение всех возможных вариантов ситуаций, в которых может оказаться агент, и реализация набора правил поведения, которые позволяют агенту достичь требуемой цели в заданных условиях. Для многих задач принятия решений в робототехнике, экономике, автоматизации процессов в промышленности существуют программные языки и пакеты библиотек, реализующие данный подход.

Обучение с учителем В некоторых случаях оказывается возможным вместо ручной разметки возможных случаев и вариантов поведения набрать так называемое множество *прецедентов* (примеров), демонстрирующих оптимальные варианты принятия решений для разных ситуаций, и применить классические методы статистического или интеллектуального машинного обучения. В случае дискретного набора действий данный подход сводится к задаче многоклассовой классификации и для случая последовательного принятия решений также называется *клонированием поведения*. Естественным образом качество поведения агента в таких случаях ограничено сверху качеством экспериментальных прецедентов.

Оптимизация Более общий подход, не предполагающий наличия демонстраций, заключается в определении пространства возможных стратегий агента и некоторой *метрики качества*, связанной с целью агента, которую необходимо максимизировать. Другими словами, задаётся *функция потерь* (некоторых ресурсов, затрачиваемых при достижении цели), которую необходимо минимизировать. Оценка качества стратегии подразумевает возможность ее запуска на множестве (пакете) *симуляций*¹ среды. Поиск оптимальной стратегии в этом пространстве происходит за счёт использования некоторого оптимизационного алгоритма. Запуск симуляции среды предполагает наличие модели динамики среды, но эта модель не всегда необходима для осуществления поиска оптимальной стратегии. В зависимости от размерности пространства стратегий и вида метрик качества могут использоваться различные алгоритмы оптимизации, как выпуклой, так и невыпуклой.

Планирование Планирование поведения представляет собой особую форму задачи оптимизации, в которой известна и используется модель динамики среды для осуществления поиска оптимальной последовательности действий для достижения цели. Автоматизированное планирование представляет собой широкий раздел методов искусственного интеллекта [282], но в основном рассматривает детерминированные постановки задач, в т.ч. и высокоразмерные. Отдельный набор методов планирования

¹Здесь и далее под симуляцией подразумевается модель среды (и виртуальной, и реальной) и процесс ее запуска, разворачивания, начиная с некоторых начальных условий.

направлен на работу в условиях неопределенности, когда детерминированная модель лишь *приближенно аппроксимирует* реальную динамику среды.

Обучение с подкреплением В обучаемой постановке задачи принятия решений не требуется заранее задавать модель динамики среды. Вместо этого агентудается возможность учить (обновлять) стратегию принятия решений в ходе процесса взаимодействия со средой так называемым «*методом проб и ошибок*» [685]. Поиск оптимальной стратегии происходит также путем максимизации метрики качества, которая выражается в виде суммы так называемых скалярных *вознаграждений* (как положительных, так и отрицательных), напрямую оценивающих качество выполненных действий. Так как процесс генерации вознаграждения чаще всего является одним из процессов среды, то получение вознаграждения может быть отложенным по времени относительно совершенного действия, повлиявшего на этот процесс в среде [149]. Действия агента, таким образом, влияют не только на получаемые наблюдения, но и на процесс своего собственного обучения, опирающегося на последовательность получаемых вознаграждений.

В настоящей работе предлагается новая методология синтеза двух подходов к поиску (суб)оптимальной стратегии последовательного принятия решений в условиях двух типов источников неопределенности. Первый тип неопределенности проистекает из неизвестности точной модели среды — как модели динамики, так и модели вознаграждения. Для возможности использования методов планирования необходимо строить аппроксимацию модели среды непосредственно в процессе взаимодействия агента со средой. Второй тип неопределенности связан с неполнотой информации, содержащейся в наблюдениях агента. Так называемая *частичная наблюдаемость* является наиболее частым случаем в практических задачах [343].

Необходимость интеграции методов автоматического планирования и машинного обучения с подкреплением связана с компенсацией комплементарных недостатков этих подходов. С одной стороны, методы планирования эффективны при наличии хороших аппроксимаций для информационных состояний и модели среды (т.е. при условиях низкой неопределенности обоих типов), т.е. позволяют достаточно быстро находить качественные решения с некоторыми гарантиями полноты и оптимальности при определенных условиях. С другой стороны, методы обучения с подкреплением не требуют полной наблюдаемости и какой-либо модели среды, однако обладают низкой эффективностью, т.е. требуют большого количества шагов взаимодействия агента со средой для обеспечения условий нахождения качественной стратегии. При этом только в очень редких, частных случаях можно говорить о некоторых гарантиях минимизации возможных потерь.

Интегрированный подход призван, с одной стороны, увеличить эффективность процесса поиска (суб)оптимальной стратегии. С другой стороны, ожидается, что в этом случае неточность и неполнота модели не окажутся непреодолимым препятствием на пути прогнозного планирования развития событий в среде, которые можно будет учитывать при выборе дальнейших действий агента, при пополнении базы прецедентов и в итоге при обновлении самой стратегии.

1.1.2 Марковский процесс принятия решений

Постановка задачи одновременного обучения и планирования опирается на определение марковского процесса принятия решений (МППР) и на базовые концепции обучения с подкреплением: оператор Беллмана [150], полезность и стратегия. Затем будет введено понятие наблюдения и его связь с состоянием. Далее перейдем к определению критика и актора. Блок определений завершим введением понятия отложенного опыта и возможных вариантов использования демонстраций, как первого приближения к неявной модели среды, выраженной в собранных прецедентах.

Здесь и далее мы предполагаем, что весь процесс взаимодействия среды и агента может быть разбит на этапы (эпизоды), каждый из которых состоит из конечного или бесконечного числа шагов, соответствующих дискретными моментам времени $t \in T$.

Определение 1.1.1 (Марковский процесс принятия решений [513]). *Марковский процесс принятия решений (МППР) определяется как пятёрка $\langle S, A, T, R, \gamma \rangle$, где:*

- S — пространство состояний системы агент-среда (например, дискретное конечное множество $\{s_1, s_2, \dots, s_n\}$);
- A — пространство действий, которые агент может совершать в среде (например, дискретное конечное множество $\{a_1, a_2, \dots, a_m\}$);
- $T : S \times A \rightarrow \Delta(S)$ — марковское ядро переходов, определяющее по текущему состоянию системы и действию распределение вероятностей на состояниях в следующий момент времени², в частном случае дискретного множества состояний — это матрица переходов;
- $R : S \times A \times S \rightarrow \mathbb{R}$ — функция вознаграждений — скаляр $r(s_t, a_t, s_{t+1})$ передаваемые агенту при переходе из состояния s_t в состояние s_{t+1} при совершении действия a_t ;
- $\gamma \in [0, 1]$ — дисконтирующий множитель, определяющий эффективный горизонт, на котором заканчивается взаимодействие агента и среды (эффективная длина эпизода).

Для удобства в дальнейшем в качестве функции вознаграждения может применяться усредненное по возможным следующим состояниям значение $r(s, a) = \mathbb{E}_{s' \sim T(s, a)} [r(s, a, s')]$. Обозначение \mathbb{E}_x будем использовать для обозначения множества или распределения x , определяющего выборку случайной величины, для которой считается матожидание. Также введем понятие *отдачи* $R(\tau)$ за эпизод взаимодействия со средой τ , которую будем определять как сумму дисконтированных вознаграждений, полученных агентом в этом эпизоде: $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$.

Будем считать, что последовательность принятых агентом решений в среде выражается в виде последовательности действий, которая, в свою очередь, задается *стратегией* агента, стохастической в общем случае.

²Здесь и далее будем обозначать через $\Delta(X)$ — множество вероятностных распределений на конечном множестве X .

Определение 1.1.2 (Стратегия агента [532]). Будем называть стратегией агента отображение $\pi : S \rightarrow \Delta(A)$, которое ставит в соответствие текущему состоянию системы некоторое распределение на множестве возможных действий.

Для удобства также будем пользоваться в качестве обозначения для вероятности выбора такого-то действия a выражением $\pi(a|s)$. Частным случаем является простая детерминированная стратегия, в которой в каждый момент времени выбирается конкретное действие $a \in A$, т.е.

$$\pi(a_t|s_t) = \begin{cases} 1, & \text{если } a_t = a, \\ 0, & \text{иначе.} \end{cases} \quad (1.1)$$

Основной задачей в машинном обучении с подкреплением является задача поиска или формирования оптимальной стратегии π^* в процессе взаимодействия со средой. Оптимальной считается такая стратегия, которая позволяет агенту достичь максимальной отдачи $R^\pi(\tau)$ за один эпизод τ взаимодействия со средой:

$$\pi^* \in \arg \max_{\pi} \mathbb{E}_{\tau, \pi} [R^\pi(\tau)] = \arg \max_{\pi} \mathbb{E}_{\substack{s_t \sim T(\cdot|s_{t-1}, a_{t-1}) \\ a_t \sim \pi(\cdot|s_t)}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \quad (1.2)$$

Здесь и далее под величиной X^π будем понимать значение X при фиксированной стратегии агента π .

Важнейшим понятием в машинном обучении с подкреплением является понятие полезности состояния $V(s)$ и полезности действия $Q(s, a)$.

Определение 1.1.3 (Полезность состояния и действия). Полезностью состояния s будем называть ожидаемое будущее вознаграждение, получаемое агентом при применении стратегии π из состояния s :

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{i=1}^{\infty} \gamma^i r(s_{t+i}, a_{t+i}) | s_t = s \right]. \quad (1.3)$$

Полезностью действия a , примененного в состоянии s , будем называть ожидаемое будущее вознаграждение получаемое агентом при использовании стратегии π для будущих состояний в эпизоде:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{i=1}^{\infty} \gamma^i r(s_{t+i}, a_{t+i}) | s_t = s, a_t = a \right]. \quad (1.4)$$

Полезность, следуя данным определениям, является минимальной моделью среды для агента, аккумулирующей в себе одновременно информацию и о возможных переходах в среде при фиксированной стратегии, и о получаемом вознаграждении. Связь между значениями полезностей состояния и действия выражается через усреднение по стратегии, в соответствии с которой выбираются действия в оцениваемом состоянии: $V^\pi(s) = \sum_{a \in A} \pi(a|s) Q^\pi(s, a)$.

Процедура подсчета полезностей состояний при фиксированной стратегии π называется оценкой данной стратегии и обычно реализуется с помощью итеративного применения

обновления полезностей по *уравнению Беллмана*. Это уравнение задает связь полезностей следующих друг за другом состояний и определяется так называемым *оператором Беллмана*:

$$\mathcal{B}^\pi[V](s) = \mathbb{E}_{\substack{s' \sim T(\cdot|s,a) \\ a \sim \pi(\cdot|s)}}[r(s,a) + \gamma V(s')]. \quad (1.5)$$

Известен следующий факт об итеративной процедуре применения оператора Беллмана для любого марковского процесса принятия решений.

Теорема 1.1.1 (О сходимости итеративной процедуры оценки стратегии [532]). *Для любого начального приближения полезности V последовательная процедура применения оператора Беллмана, определенного в 1.5 для фиксированной стратегии π , сходится к истинному значению полезности состояний для стратегии π :*

$$\lim_{n \rightarrow \infty} (\mathcal{B}^\pi)^n[V] = V^\pi. \quad (1.6)$$

Данная теорема является прямым следствием того факта, что оператор Беллмана является сжимающим оператором с единственной неподвижной точкой. Оператор Беллмана для функции полезности действия Q записывается аналогичным образом.

Обобщенный итеративный алгоритм оценки стратегии (AMPI) [514] с использованием оператора Беллмана выглядит следующим образом:

$$\begin{cases} \pi_{k+1} \in \mathcal{G}(Q_k), \\ Q_{k+1} = (\mathcal{B}^{\pi_{k+1}})^m Q_k + \varepsilon_{k+1}, \end{cases} \quad (1.7)$$

где

- $\mathcal{G}(Q) = \arg \max_\pi \sum_a \pi(s,a) Q(s,a)$ — множество жадных по функции полезности стратегий,
- ε отвечает за ошибку применения оператора Беллмана.

Завершая постановку задачи машинного обучения с подкреплением, необходимо отметить, что агенту неизвестны ни функция вознаграждения R , ни функция переходов системы T . Т.е. методы, которые не занимаются приближенной оценкой этих функций, относятся к классу безмодельных методов. В дальнейшем будет рассматриваться класс методов, в которых в явном виде восстанавливается модель через аппроксимацию функций R и T .

Также необходимо отметить, что во многих средах условие полной наблюдаемости среды не выполняется. Это означает, что агент не имеет непосредственного доступа к информационному состоянию s_t в каждый момент времени, а получает от среды только так называемое *наблюдение* $o_t \in O$. Последовательность прецедентов $o_1, a_1, r_2, o_2, a_2, r_3, \dots$ уже не будет являться марковским процессом в связи с тем, что информации в наблюдении o_t не достаточно для принятия оптимального решения a_t . Формально такой процесс называется *частично наблюдаемым марковским процессом* [343].

Стандартной практикой в этом случае является введение некоторой функции $s_t \approx h(o_t, o_{t-1}, \dots)$ от истории наблюдений (полной или с некоторым горизонтом), которая аппроксимирует действительное информационное состояние s_t . В качестве

таких аппроксиматоров могут выступать в том числе и рекуррентные нейронные сети, параметры которых могут настраиваться в ходе взаимодействия агента со средой, позволяя подстраивать необходимую глубину истории наблюдений. В дальнейшем мы будем предполагать, что такая функция h в случае сред с частичной наблюдаемостью задана и является частью архитектуры агента. Без потери общности будем далее везде использовать обозначение s_t для информационного состояния, даже если оно получено в результате работы аппроксиматора, а агенту доступно только наблюдение o_t .

1.1.3 Обучение на основе функции полезности

Определенная выше функция полезности состояния или действия представляет собой концентрированную информацию о возможных переходах в среде и получаемых агентом вознаграждениях. На основе функции полезности для данной стратегии агент может предсказывать развитие событий в МППР и таким образом выбирать оптимальные действия. Для этого необходимо ввести понятие оптимальной функции полезности.

Определение 1.1.4 (Оптимальная функция полезности). *Оптимальная функция полезности состояний $V^*(s)$ — это максимальное значение функции полезности по всем стратегиям:*

$$V^*(s) = \max_{\pi} V^{\pi}(s).$$

Оптимальная функция полезности действий $Q^(s,a)$ — это максимальное значение функции полезности по всем стратегиям:*

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a).$$

Если агенту доступна оптимальная функция полезности Q^* , то из нее сразу же следует и оптимальная детерминированная стратегия для любого марковского процесса принятия решений:

$$\pi^*(a|s) = \begin{cases} 1, & \text{если } a = \arg \max_{a \in A} Q^*(s,a), \\ 0, & \text{иначе.} \end{cases}$$

Справедливо и более общее утверждение.

Теорема 1.1.2 (Об оптимальной стратегии [532]). *Для любого марковского процесса принятия решений*

- существует оптимальная стратегия π^* , которая лучше всех других стратегий или эквивалентна им: $\pi^* \geq \pi, \forall \pi$,
- все оптимальные стратегии удовлетворяют оптимальной функции полезности состояний: $V^{\pi^*}(s) = V^*(s)$,
- все оптимальные стратегии удовлетворяют оптимальной функции полезности действий: $Q^{\pi^*}(s,a) = Q^*(s,a)$.

Оптимальные значения полезностей связаны следующим соотношением: $V^*(s) = \max_{a \in A} Q^*(s, a)$, что приводит нас к следующему виду оператора оптимальности Беллмана:

$$\mathcal{B}^{*,\pi}[Q](s) = \mathbb{E}_{\substack{s' \sim T(\cdot|s,a) \\ a \sim \pi(\cdot|s)}} \left[r(s, a) + \gamma \max_{a' \in A} Q(s', a') \right]. \quad (1.8)$$

Данный оператор существенно нелинеен, в связи с чем существуют только приближенные итерационные методы нахождения неподвижной точки. Решение этого уравнения гарантирует нам нахождение как оптимальной функции полезности, так и оптимальной стратегии для данного МППР.

Одним из наиболее значимых итерационных методов поиска функции полезности является *метод временных различий* (TD), в котором значение полезности обновляется на основе ожидаемой отдачи при некоторой стратегии π :

$$Q_{k+1}(s, a) = Q_k(s, a) + \alpha (\mathcal{B}^\pi[Q_k](s', a') - Q_k(s, a)),$$

где s, a — текущая пара состояние-действие, а s', a' — в следующий момент времени. Результат применения оператора Беллмана к функции Q в данном случае называется *TD-показателем*.

Итеративный алгоритм управления (SARSA) выглядит следующим образом:

$$\begin{cases} \pi_{k+1} \in \mathcal{G}^\epsilon(Q_k), \\ Q_{k+1} = Q_k + \alpha (\mathcal{B}^{\pi_{k+1}} Q_k - Q_k), \end{cases} \quad (1.9)$$

где $\mathcal{G}^\epsilon(Q)$ — множество ϵ -«жадных» по функции полезности стратегий, т.е. таких, в которых с некоторой вероятностью ϵ выбирается случайное действие, а в остальных случаях — «жадное» по функции полезности.

Применение оператора оптимальности Беллмана приводит нас к другой вариации алгоритма итерации по стратегиям с использованием *отложенного опыта* (Q-обучение):

$$\begin{cases} \pi_{k+1} \in \mathcal{G}^\epsilon(Q_k), \\ Q_{k+1} = Q_k + \alpha (\mathcal{B}^{*,\pi_{k+1}} Q_k - Q_k), \end{cases} \quad (1.10)$$

Для большого пространства состояний, например, в таких средах как навигация по изображению или игра в го, необходимо вводить аппроксимацию функции полезности для возможности оценки полезности для тех состояний, которые не входили в множество прецедентов (см. рисунок 1.2). Пусть задано множество параметров $\{\theta_1, \dots, \theta_n\}$; конкретные значения этих параметров образуют вектор θ . Введем параметрическое семейство функций полезности:

$$\hat{V}(s; \theta) \approx V^\pi(s),$$

$$\hat{Q}(s, a; \theta) \approx Q^\pi(s, a).$$

Различные варианты задания параметрических семейств функций:

Табличный: таблица с записями для каждого состояния МППР.

Агрегатор: разделение пространства состояний (или наблюдений) на множество классов.

Линейный: использование фиксированной карты признаков $x : S \rightarrow \mathbb{R}^n$ и их линейной комбинации для кодирования полезности $\hat{V}(s; \theta) = \theta^\top x(s)$.

Дифференцируемый: $\hat{V}(s; \theta)$ — (нелинейная) дифференцируемая функция от θ , карта признаков формируется в процессе обучения.

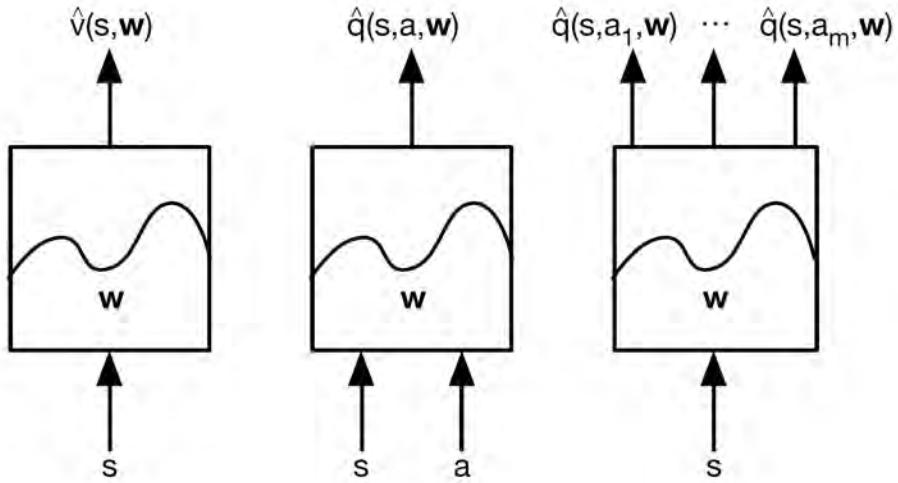


Рисунок 1.2 — Виды аппроксимации функции полезности: слева отображение вида $S \rightarrow \mathbb{R}$, посередине — вида $S \times A \rightarrow \mathbb{R}$ и справа — вида $S \rightarrow \mathbb{R}^m$.

В дальнейшем в данной работе будет подразумеваться именно дифференцируемый вариант аппроксиматоров, а именно их нейросетевые реализации. Дифференцируемость предполагает возможность вычисления градиента $\nabla_\theta \hat{V}(s_t; \theta)$, который используется в различных вариантах стохастического градиентного спуска как одного из наиболее часто используемых оптимизаторов.

Основанные на полезности методы обучения с подкреплением по сути решают задачу регрессии: выбирают скалярную функцию $\hat{V}^\pi(s_t; \theta)$ из параметризованного семейства аппроксиматоров, близкую к некоторому истинному виду функции полезности $V^\pi(s_t)$. В данном случае простейший вид функции потерь представляет собой среднеквадратичную ошибку:

$$\mathcal{L}(\theta) = \mathbb{E}_\pi \left[(V^\pi(s_t) - \hat{V}^\pi(s_t; \theta))^2 \right].$$

Так как истинное значение функции $V^\pi(s_t)$ в процессе обучения не известно, то оно заменяется на оценку, полученную, например, методом временных различий (TD):

$$\mathcal{L}(\theta) = \mathbb{E}_\pi \left[(\mathcal{B}^\pi \hat{V}(\theta) - \hat{V}(\theta))^2 \right]. \quad (1.11)$$

Отсюда следует, что изменения параметров θ в процессе градиентного спуска вычисляются следующим образом:

$$\Delta\theta = \alpha \left(\mathcal{B}^\pi \hat{V}(\theta) - \hat{V}(\theta) \right) \nabla_\theta \hat{V}(s_t; \theta),$$

где α — стандартный параметр шага градиентного спуска (параметр *скорости обучения*).

Функция потерь 1.11 в виде использования в ней оператора Беллмана обладает следующими особенностями, которые являются отличительной чертой оптимизационных задач в обучении с подкреплением:

Скоррелированность данных Матожидание в функции потерь берется по множеству прецедентов, которые получены вследствие применения некоторой стратегии агента и в результате последовательного применения действий агента в среде. Это означает, что последовательные прецеденты существенно скоррелированы. В связи с этим получаемый опыт агента не состоит из независимых одинаково распределенных значений некоторой случайной величины V .

Нестационарность целевого значения В отличие от классической постановки, в данной функции потерь целевое значение регрессии является нестационарным вследствие как и меняющей стратегии, так и за счет возможной нестационарности динамики среды (например, из-за наличия других обучающихся агентов). Также на нестационарность целевого значения влияет невыполнение условия полной наблюдаемости, что чаще всего и происходит в реальных задачах.

Обобщенный алгоритм итерации по стратегиям с использованием аппроксиматоров функции полезности выглядит аналогично алгоритму Q-обучения 1.10:

$$\begin{cases} \pi_{k+1} \in \mathcal{G}^\varepsilon(\hat{Q}(\theta_k)), \\ \theta_{k+1} = \theta_k + \alpha \left(\mathcal{B}^{\star, \pi_{k+1}} \hat{Q}(\theta_k) - \hat{Q}(\theta_k) \right) \nabla_\theta \hat{Q}(\theta_k). \end{cases} \quad (1.12)$$

Описанные выше особенности постановки оптимизационной задачи обычно преодолеваются путем использования двух техник: формирование памяти прецедентов (replay buffer) и фиксация TD-показателя при расчете целевого значения.

Для борьбы со скоррелированностью данных при расчете градиентов по мини-выборке (minibatch) введем память прецедентов \mathcal{D} , куда будут сохраняться все четверки вида (s_t, a_t, r_{t+1}, s_t) после каждого выполнения действия агента в среде. Обычно размер памяти прецедентов фиксируется, и используется некоторая техника вытеснения более старых по времени прецедентов новыми. Мини-выборка, по которой считается несколько последовательных обновлений весов $\Delta\theta$, формируется путем случайной выборки (семплирования) из памяти прецедентов. Именно случайное расположение прецедентов во времени при их генерации в среде позволяет существенно снизить эффект скоррелированности данных.

Фиксация TD-показателя работает следующим образом. Введем отдельное множество параметров θ^- . По заранее определенному расписанию (например, раз в n шагов обновления) текущие параметры θ_k копируются в параметры θ^- . При вычислении TD-показателя в процессе Q-обучения будем использовать именно фиксированные параметры θ^- :

$$\mathcal{L}_k(\theta) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[(\mathcal{B}^\pi \hat{V}(\theta_k^-) - \hat{V}(\theta_k))^2 \right]. \quad (1.13)$$

За счет более редкого обновления параметров, по которым считается целевое значение задачи регрессии, снижается эффект нестационарности целевого значения и свойства сходимости процесса обучения существенно улучшаются.

При использовании в качестве аппроксиматоров семейства параметрических композиционных нелинейных преобразований (нейронных сетей с полносвязными,

сверточными, трансформерными слоями) функция полезности приобретает характерный композиционный вид:

$$\hat{Q}(s, a; \theta) = h_n(h_{n-1}(\dots h_1(s, a))),$$

где $h_i = g_i(\theta^i h_{i-1})$ — это комбинация линейного по подмножеству параметров $\theta^i \in \theta$ преобразования выходов h_{i-1} и некоторого нелинейного преобразования g_i (активационной функции). Такое композиционное представление выгодно во многих случаях за счет известного цепного правила вычисления градиента такой функции последовательно по подмножествам параметров $\theta^1, \theta^2, \dots, \theta^n$.

Описанный выше метод Q-обучения с использованием сверточного нейросетевого аппроксиматора, памяти предшествующих действий и фиксации TD-показателя получил название глубокой Q-сети (DQN) [323] и продемонстрировал первые значимые результаты в обучении стратегии агента в средах с наблюдениями, задаваемыми изображениями (трехмерными тензорами высокой размерности) (см. рисунок 1.3). Дальнейшие улучшения алгоритма DQN (такие как двойной DQN [312], дуэльный DQN [239], алгоритма радуги (Rainbow) [518]) в некоторой степени улучшили полученные результаты, но не поменяли базовых принципов обучения на основе функции полезности.

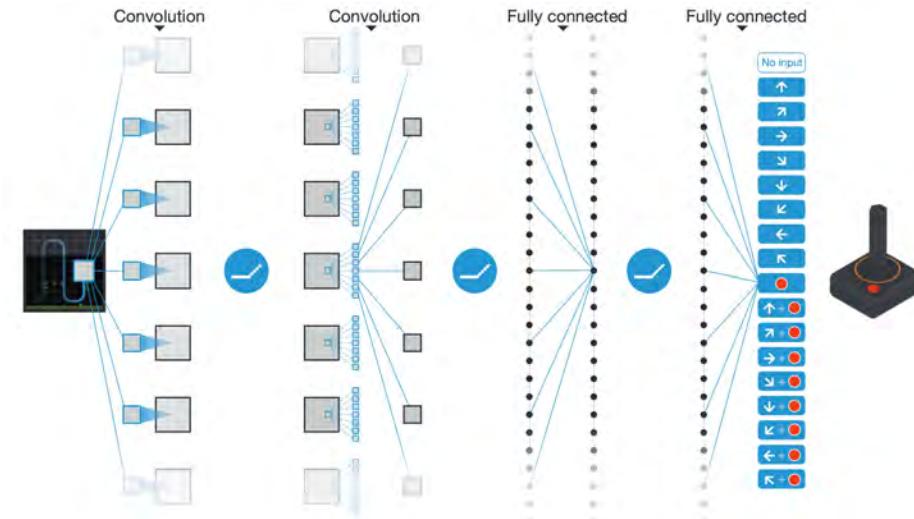


Рисунок 1.3 — Обработка высокоразмерного наблюдения в алгоритме глубокой Q-сети (DQN) [323] с помощью сверточной нейронной сети.

1.1.4 Параметризация и градиент стратегии

Рассмотренные выше методы, основанные на функции полезности, предполагают формирование детерминированной, «жадной» по полезности стратегии. В случае частично наблюдаемой среды часто оказывается, что оптимальной является стохастическая стратегия, по которой можно выбирать для каждого состояния не всегда одинаковые действия в

различные моменты времени. Использование стохастической стратегии также обосновано тем, что сразу же включает в себя возможность исследования среды без введения дополнительных эвристических соображений, как, например, в ϵ -«жадной» стратегии.

Для возможности работы со стохастическими стратегиями промежуточный шаг с построением функции полезности оказывается излишним, и поэтому обычно вводится прямая параметризация функции стратегии в виде $\hat{\pi}(s; \varphi) = \mathbb{P}[a|s, \varphi]$, где $\mathbb{P}[a] \in \Delta(A)$ — некоторое распределение на множестве действий (не обязательно дискретных).

Определение 1.1.5 (Траектория (опыта агента)). *Будем называть траекторией следующую последовательность состояний-действий длины T , которые агент получает в процессе взаимодействия со средой:*

$$\tau = (s_0, a_0, r_0, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T).$$

Также введем понятие *отдачи траектории* без дисконтирования, то есть обычной суммы вознаграждений по всей траектории $\hat{R}(\tau) = \sum_{t=0}^T r(s_t, a_t)$.

Для прямого поиска оптимальной стратегии в пространстве параметров φ необходимо ввести так называемую *функцию полезности стратегии J* . Существует несколько вариантов ее задания:

Начальная полезность В эпизодических средах есть возможность работать с полными траекториями и оценивать качество стратегии по полезности первого состояния в траектории τ :

$$J_1(\varphi) = V^{\hat{\pi}(\varphi)}(s_1) = \mathbb{E}_{\hat{\pi}(\varphi)}[R(\tau)]. \quad (1.14)$$

Средняя полезность Если у агента есть доступ к большому количеству прецедентов опыта, то при оценке вероятности встретить то или иное состояние можно рассчитать среднюю полезность по всем состояниям, а не только начальному:

$$J_{avV}(\varphi) = \sum_s d^{\hat{\pi}(\varphi)}(s) V^{\hat{\pi}(\varphi)},$$

где $d^{\hat{\pi}(\varphi)}(s)$ — стационарное распределение марковской цепи для стратегии $\hat{\pi}(\varphi)$.

Среднее вознаграждение В некоторых случаях оказывается удобнее отказаться от коэффициента дисконтирования, отвечающего за горизонт предсказания, и считать распределение вознаграждений по всей траектории:

$$J_{avR}(\varphi) = \sum_s d^{\hat{\pi}(\varphi)}(s) \sum_a \hat{\pi}(\varphi)r(s, a) = \mathbb{E}_{\hat{\pi}(\varphi)}[r(s, a)]. \quad (1.15)$$

В методах градиента стратегии задача поиска оптимальной стратегии является классической оптимизационной задачей: необходимо найти такое значение параметров φ , которое максимизирует выбранный функционал качества, например, $J_{avR}(\varphi)$. Решать данную оптимизационную задачу можно как неградиентными методами (метод прямого восхождения, симплекс метод или метод Нелдера-Мида, генетические алгоритмы), так и

с использованием различных вариаций стохастического градиентного спуска, который в основном и будет использоваться в данной работе.

Рассмотрим функционал качества стратегии π в виде среднего вознаграждения 1.15 и представим его в виде суммы отдач отдельных траекторий, которые генерируются по данной стратегии π :

$$J(\varphi) = J_{avR}(\varphi) = \mathbb{E}_{\hat{\pi}(\varphi)} \left[\sum_{t=0}^T r(s_t, a_t) \right] = \sum_{\tau} p(\tau; \varphi) \hat{R}(\tau), \quad (1.16)$$

где $p(\tau; \varphi)$ — распределение вероятностей по траекториям τ , полученным по стратегии $\hat{\pi}(\varphi)$.

Оптимизационная задача для поиска оптимальной стратегии запишется следующим образом:

$$\arg \max_{\varphi} J(\varphi) = \arg \max_{\varphi} \sum_{\tau} p(\tau; \varphi) \hat{R}(\tau). \quad (1.17)$$

Для решения данной задачи с помощью градиентных методов выпишем выражение для градиента функции полезности стратегии:

$$\begin{aligned} \nabla_{\varphi} J(\varphi) &= \nabla_{\varphi} \sum_{\tau} p(\tau; \varphi) \hat{R}(\tau) = \\ &= \sum_{\tau} \nabla_{\varphi} p(\tau; \varphi) \hat{R}(\tau) = \sum_{\tau} \frac{p(\tau; \varphi)}{p(\tau; \varphi)} \nabla_{\varphi} p(\tau; \varphi) \hat{R}(\tau) = \\ &= \sum_{\tau} p(\tau; \varphi) \hat{R}(\tau) \frac{\nabla_{\varphi} p(\tau; \varphi)}{p(\tau; \varphi)} = \sum_{\tau} p(\tau; \varphi) \hat{R}(\tau) \nabla_{\varphi} \log p(\tau; \varphi). \end{aligned}$$

Введение логарифма стратегии позволило нам записать выражение градиента полезности стратегии через функцию от самого значения полезности. Таким образом мы сохранили возможность расчета матожидания по распределению на множестве траекторий. Данный прием называется введением логарифмической производной и позволяет ввести понятие оценки *максимального правдоподобия* для стратегии (см. рисунок 1.4).

В стохастическом случае с использованием оценки градиента \hat{g} по выборке размером m получим следующее эмпирическое выражение:

$$\nabla_{\varphi} J(\varphi) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^m \hat{R}(\tau^{(i)}) \nabla_{\varphi} \log p(\tau^{(i)}; \varphi).$$

Теперь распишем данное выражение, заменив вероятность $p(\tau^{(i)}; \varphi)$ появления траектории $\tau^{(i)}$ через вероятность появление отдельных состояний в этой траектории. Пусть $\mu(s_0)$ — распределение начальных состояний; тогда

$$\begin{aligned} \nabla_{\varphi} \log p(\tau^{(i)}; \varphi) &= \nabla_{\varphi} \log \left[\mu(s_0) \prod_{t=0}^{T-1} \hat{\pi}(s_t; \varphi) T(s_{t+1}|s_t, a_t) \right] = \\ &= \nabla_{\varphi} \left[\log \mu(s_0) + \sum_{t=0}^{T-1} \log \hat{\pi}(s_t; \varphi) + \log T(s_{t+1}|s_t, a_t) \right] = \\ &= \sum_{t=0}^{T-1} \nabla_{\varphi} \log \hat{\pi}(s_t; \varphi). \end{aligned}$$

При вычислении градиента функции полезности стратегии зависимость от конкретных переходов в самой среде исчезла и осталась только зависимость от совершаемых на протяжении всей траектории действий, распределение которых задается самой функцией стратегии. Данное выражение показывает независимость градиента функции полезности стратегии от динамики среды, и в стохастическом случае мы получаем следующее выражение:

$$\nabla_{\varphi} J(\varphi) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^m \hat{R}(\tau) \nabla_{\varphi} \log p(\tau^{(i)}; \varphi) = \quad (1.18)$$

$$= \frac{1}{m} \sum_{i=1}^m \hat{R}(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_{\varphi} \log \pi(s_t^{(i)}; \varphi). \quad (1.19)$$

Данная оценка градиента является несмещенной, но достаточно шумной.

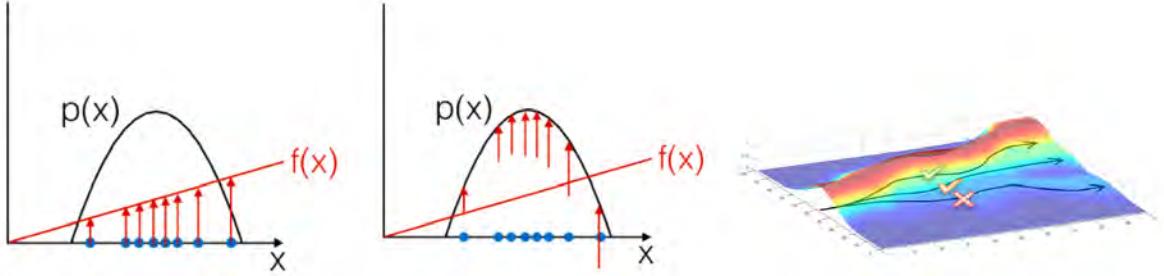


Рисунок 1.4 — Визуальная интерпретация отношения правдоподобия при обновлении параметров стратегии в соответствии с полезностью траектории опыта.

Обобщением полученного выражения для градиента функции полезности стратегии на другие виды функции полезности и, таким образом, на неэпизодические среды является теорема о градиенте стратегии, где роль отдачи траектории играет истинная оценка полезности действий $Q^{\hat{\pi}}$ для аппроксимации стратегии $\hat{\pi}$.

Теорема 1.1.3 (Теорема о градиенте стратегии [649]). Для любой дифференцируемой стратегии $\hat{\pi}(s; \varphi)$, для любой функции полезности стратегии $J = J_1, J_{avR}$ или $\frac{1}{1-\gamma} J_{avV}$ градиент стратегии равен:

$$\nabla_{\varphi} J(\varphi) = \mathbb{E}_{\hat{\pi}(\varphi)} [Q^{\hat{\pi}(\varphi)}(s, a) \nabla_{\varphi} \log \hat{\pi}(a; \varphi)].$$

Однако, как и в случае с методами, основанными на полезности, прямое использование данной теоремы невозможно, так как истинное значение функции полезности действия $Q^{\hat{\pi}(\varphi)}$ не известно. В этом случае мы можем заменять это значение на приближенное значение, вычисленное с использованием Монте-Карло оценки или по методу временных различий. Первый вариант с заменой полезности на отдачу $R_t(\tau)$, посчитанную от состояния s_t до конца эпизода τ длины T :

$$\begin{cases} \tau, T \sim \hat{\pi}(\varphi_k), \\ \varphi_{k+1} = \varphi_k + \alpha \sum_{t=1}^{T-1} \nabla_{\varphi} \log \hat{\pi}(s_t; \varphi_k) R_t(\tau). \end{cases} \quad (1.20)$$

Представленный выше алгоритм называется REINFORCE и является базовым подходом, на основе которого строятся уже другие, более современные алгоритмы.

Рассмотрим примеры параметризованных семейств функций стратегии, из которых обычно выбираются аппроксиматоры:

Логистические стратегии Рассмотрим вектор характерных признаков $f(s,a)$ пары состояние-действие; тогда на основе него можно рассмотреть экспоненциальное усреднение по данным признакам для кодирования стратегии:

$$\hat{\pi}(s; \varphi) = \frac{e^{f(s,a)^\top \varphi}}{\sum_a e^{f(s,a)^\top \varphi}}$$

Функция вклада в градиент выглядит следующим образом:

$$\nabla_\varphi \log \hat{\pi}(s; \varphi) = f(s,a) - \mathbb{E}_{\hat{\pi}(\varphi)}[f(s, \cdot)].$$

Гауссовые стратегии Для непрерывного пространства действий стандартным кодировщиком считается гауссово распределение $\mathcal{N}(\mu(s), \sigma^2)$, дисперсия σ^2 которого может быть зафиксирована, а среднее кодироваться набором признаком $\mu(s) = f(s)^\top \varphi$. Тогда функция вклада примет следующий хорошо интерпретируемый вид:

$$\nabla_\varphi \log \hat{\pi}(s; \varphi) = \frac{(a - \mu(s))f(s)}{\sigma^2}.$$

1.1.5 Методы «актора-критика»

Представленный в алгоритме 1.20 метод REINFORCE обладает существенным недостатком, связанным с тем, что используемая в нем оценка градиента 1.18, хотя и является несмещенной, обладает очень высокой дисперсией. Основными подходами в борьбе с этой дисперсией в методах градиента стратегии являются введение базового уровня оценки и использование других методов оценки, помимо Монте-Карло. Первый подход приводит к использованию функции преимущества для оценки полезности действия, а второй метод — к введению отдельной параметризации для функции полезности. Совмещение этих подходов в единую архитектуру образует семейство алгоритмов «актор-критик».

Стандартной практикой для подавления дисперсии некоторой случайной является вычитание эмпирической оценки ее матожидания. В случае с методами градиента стратегии это будет выглядеть следующим образом:

$$\nabla_\varphi \mathbb{E}_\tau[\hat{R}(\tau)] = \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \nabla_\varphi \log \pi(a_t; \varphi) \left(\sum_{t'=t}^{T-1} r_{t'} - B(s_t) \right) \right],$$

где базовый уровень $B(s_t)$ не зависит от выбранных действий и является эмпирической оценкой ожидаемой отдачи.

Введение базового уровня не вносит смещенностей к оценке градиента:

$$\begin{aligned}
& \mathbb{E}_\tau [\nabla_\varphi \log \hat{\pi}(a_t; \varphi) B(s_t)] = \\
&= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} [\mathbb{E}_{s_{(t+1):T}, a_{t:(T-1)}} [\nabla_\varphi \log \hat{\pi}(a_t; \varphi) B(s_t)]] = \\
&= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} [B(s_t) \mathbb{E}_{s_{(t+1):T}, a_{t:(T-1)}} [\nabla_\varphi \log \hat{\pi}(a_t; \varphi)]] = \\
&= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} [B(s_t) \mathbb{E}_{a_t} [\nabla_\varphi \log \hat{\pi}(a_t; \varphi)]] = \\
&= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[B(s_t) \sum_a \hat{\pi}(a_t; \varphi) \frac{\nabla_\varphi \hat{\pi}(a_t; \varphi)}{\hat{\pi}(a_t; \varphi)} \right] = \\
&= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[B(s_t) \sum_a \nabla_\varphi \hat{\pi}(a_t; \varphi) \right] = \\
&= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} \left[B(s_t) \nabla_\varphi \sum_a \hat{\pi}(a_t; \varphi) \right] = \\
&= \mathbb{E}_{s_{0:t}, a_{0:(t-1)}} [B(s_t) \nabla_\varphi 1] = 0.
\end{aligned}$$

С учетом базового уровня, который можно обновлять в ходе обучения агента, получается следующая модификация алгоритма REINFORCE, которая называется базовым градиентом стратегии (vanilla PG):

$$\begin{cases} \{\tau_i, T_i\} \sim \hat{\pi}(\varphi_k), \\ \varphi_{k+1} = \varphi_k + \alpha \sum_i \sum_{t=1}^{T_i-1} \nabla_\varphi \log \hat{\pi}(s_t; \varphi_k) (R_t(\tau_i) - B_k(s_t)) \\ B_{k+1}(s_t) = \arg \min_{B_k} \sum_i \sum_t \|R_t(\tau) - B_k(s_t)\|. \end{cases} \quad (1.21)$$

Используемая в базовом градиенте стратегии величина $\hat{A}(a_t, s_t) = R_t(\tau) - B_k(s_t)$ называется оценкой *функции преимущества* и интерпретируется как дополнительное вознаграждение, которое может дать выбор именно действия a_t по сравнению со средним значением, которое оценивается через $B(s_t)$. В более общем виде функция преимущества определяется следующим образом.

Определение 1.1.6 (Функция преимущества). *Функцией преимущества действия A^π при фиксированной стратегии π будем называть разность функции полезности действия Q^π и функции полезности состояния V^π :*

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) = Q^\pi(s, a) - \mathbb{E}_\pi [Q^\pi(s_t, a_t)].$$

Второй подход к уменьшению дисперсии для оценки градиента стратегии состоит в введении отдельного аппроксиматора функции полезности $Q^{\hat{\pi}(\varphi)}$, которая фигурирует в теореме о градиенте стратегии 1.1.3:

$$\hat{Q}(s, a; \theta) \approx Q^{\hat{\pi}(\varphi)}(s, a).$$

Отдельный аппроксиматор с параметрами θ называется *критиком* и выполняет функцию оценки текущей стратегии, которая обновляется на основе градиента стратегии в модуле *актор*. Таким образом, архитектура агента содержит два модуля (см. рисунок 1.5):

критик обновляет параметры θ функции полезности действия и оценивает текущую стратегию, генерируемую актором,
актор обновляет параметры φ стратегии с учетом предположений критика о полезности тех или иных действий.

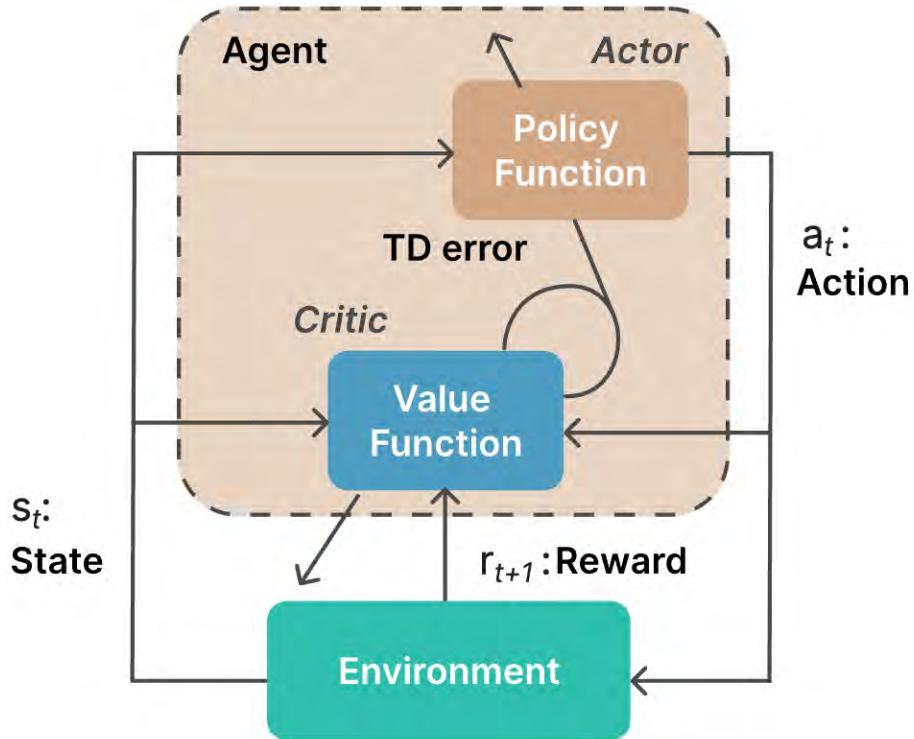


Рисунок 1.5 — Общая схема работы архитектуры «актор-критик»: обучение критика (Critic) по TD-ошибке функции полезности (Value Function) и обновление актора (Actor) с функцией стратегии (Policy Function).

Пусть критик обновляется по методу временных различий $TD(0)$ и использует простую линейную аппроксимацию функции полезности по признакам $f(s,a)$; тогда базовый алгоритм «актор-критик» (QAC) выглядит следующим образом:

$$\begin{cases} \varphi_{k+1} = \varphi_k + \alpha \nabla_\varphi \log \hat{\pi}(\varphi_k) Q(\theta_k) \\ \theta_{k+1} = \theta_k + \beta (\mathcal{B}^{\hat{\pi}(\varphi_k)} \hat{Q}(\theta_k) - \hat{Q}(\theta_k)) f. \end{cases} \quad (1.22)$$

Данный алгоритм естественным образом обобщается на любую функцию аппроксимации критика и на другие виды оператора Беллмана. Также возможно использование функции преимущества вместо функции полезности. При этом нет необходимости вводить третье множество параметров для функции полезности состояния. Воспользуемся следующим свойством матожидания истинной TD- ошибки $\delta^{\hat{\pi}(\varphi)}$:

$$\begin{aligned} \mathbb{E}_{\hat{\pi}(\varphi)}[\delta^{\hat{\pi}(\varphi)}|s,a] &= \mathbb{E}_{\hat{\pi}(\varphi)}[r + \gamma V^{\hat{\pi}(\varphi)}(s')|s,a] - V^{\hat{\pi}(\varphi)}(s) \\ &= Q^{\hat{\pi}(\varphi)}(s,a) - V^{\hat{\pi}(\varphi)}(s) \\ &= A^{\hat{\pi}(\varphi)}(s,a). \end{aligned}$$

С учетом этого факта получается следующий вариант алгоритма — «актор-критик» с преимуществом (A2C):

$$\begin{cases} \delta(\theta_k) = \mathcal{B}^{\hat{\pi}(\varphi_k)} \hat{V}(\theta_k) - \hat{V}(\theta_k) \\ \varphi_{k+1} = \varphi_k + \alpha \nabla_\varphi \log \hat{\pi}(\varphi_k) \delta(\theta_k) \\ \theta_{k+1} = \theta_k + \beta \delta(\theta_k) \nabla_\theta V(\theta_k). \end{cases} \quad (1.23)$$

В связи с тем, что метод временных различий привносит дополнительное смещение к оценке полезности, необходимо с осторожностью выбирать виды аппроксиматоров и методы обновления параметров для критика.

Так как аппроксимация, используемая в критике, приводит к нарушению условий теоремы о градиенте стратегии 1.1.3, необходимо заново переопределить условий для подсчета градиента стратегии. Справедлива следующая теорема, которая накладывает ограничения на вид аппроксимации функции полезности и тем самым сохраняет то же самое выражение для актора:

Теорема 1.1.4 (о совместимой функции аппроксимации [502]). *Если удовлетворены следующие два условия:*

1. *Аппроксиматор функции полезности **совместим** со стратегией:*

$$\nabla_\theta Q(s, a; \theta) = \nabla_\varphi \log \pi(s; \varphi).$$

2. *Параметры функции полезности минимизируют среднеквадратичную ошибку:*

$$\varepsilon = \mathbb{E}_{\pi(\varphi)}[(Q^{\pi(\varphi)}(s, a) - Q(s, a; \varphi))^2],$$

тогда градиент стратегии в точности равен

$$\nabla_\varphi J(\varphi) = \mathbb{E}_{\pi(\varphi)}[\nabla_\varphi \log \pi(s; \varphi) Q(s, a; \theta)].$$

Практическим следствием данной теоремы являются рекомендации к выбору модели для критика: использовать простые архитектуры аппроксиматора (полносвязные слои, если речь идет о нейросетевых моделях), чтобы обеспечить первое условие теоремы и только квадратичную функцию потерь с приемами уменьшения нестационарности целевой переменной (например, с фиксацией весов аппроксиматора) для удовлетворения второго условия.

1.1.6 Оптимизация градиента стратегии

В методах градиента стратегии важно настроить процесс градиентного спуска таким образом, чтобы на каждом шаге обновления параметров полезность новой стратегии $\hat{\pi}(\varphi')$ была больше или равна полезности предыдущей версии стратегии $\hat{\pi}(\varphi)$: $J(\hat{\pi}(\varphi')) \geq J(\hat{\pi}(\varphi))$.

Как и в классическом машинном обучении, в котором также решается задача поиска оптимумов некоторой функции потерь, размер шага градиентного спуска, который регулируется параметром скорости обучения, играет большую роль при поиске стратегии агента. Однако ситуация осложняется тем, что в обучении с учителем, например, слишком большой шаг в направлении градиента может быть достаточно легко исправлен на следующей итерации обновления параметров. В обучении с подкреплением из-за нестационарности процесса обучения и наличия обратной связи от среды большое изменение параметров стратегии в неверном направлении будет существенно труднее исправить. Это связано с тем, что неудачное изменение параметров приводит к неэффективной стратегии, которая с малой вероятностью приведет к получению новых полезных данных, а это, в свою очередь, приведет к тому, что новое обновление параметров будет еще хуже. Такой подкрепляющий сам себя процесс деградации производительности является известной проблемой методов градиента стратегии типа REINFORCE.

Для поиска способа гарантированного улучшения стратегии рассмотрим задачу подбора шага градиентного спуска. Также мы продолжаем работать в условиях явной параметризации стратегии и будем использовать функционал качества начальной полезности 1.14. Наша задача — предсказать полезность новой стратегии в то время, когда мы можем контролировать только выборку данных по текущей стратегии $\hat{\pi}(\varphi)$. По сути, это является задачей обучения на основе отложенного опыта, то есть оценки новой стратегии по данным, собранным старой стратегией.

Выразим ожидаемую отдачу новой стратегии в терминах функции преимущества исходной стратегии (для упрощения будем опускать)

$$J(\varphi') = J(\varphi) + \mathbb{E}_{\hat{\pi}(\varphi')} \left[\sum_{t=0}^{\infty} \gamma^t A^{\hat{\pi}(\varphi)}(a_t, s_t) \right] = J(\varphi) + \sum_s \mu_{\hat{\pi}(\varphi')}(s) \sum_a \hat{\pi}(s; \varphi') A^{\hat{\pi}(\varphi)}(s, a),$$

где $\mu_{\hat{\pi}(\varphi')}(s)$ определяется как дисконтированная частота состояний s по опыту, собранному стратегией $\hat{\pi}(\varphi')$. В данном выражении на основе текущего опыта могут быть посчитаны $A^{\hat{\pi}(\varphi)}(s, a)$ и $\hat{\pi}(\varphi')$, но значение $\mu_{\hat{\pi}(\varphi')}(s)$ не известно и его необходимо будет оценить через имеющуюся выборку.

Введем суррогатную функцию полезности стратегии, в которой значение $\mu_{\hat{\pi}(\varphi')}(s)$ заменено на вычислимое значение $\hat{\mu}_{\hat{\pi}(\varphi)}(s)$:

$$L_{\hat{\pi}}(\hat{\pi}') = J(\varphi) + \sum_s \hat{\mu}_{\hat{\pi}(\varphi)}(s) \sum_a \hat{\pi}(s; \varphi') A^{\hat{\pi}(\varphi)}(s, a)$$

Данная суррогатная функция обладает полезными свойствами. Во-первых, эти функции совпадают в начальной точке градиентного спуска: $L_{\hat{\pi}(\varphi_0)}(\hat{\pi}(\varphi_0)) = J(\varphi_0)$. Во-вторых градиент новой функции равен градиенту функции полезности стратегии, вычисленной в точке φ_0 :

$$\nabla_{\varphi} L_{\hat{\pi}(\varphi_0)}(\hat{\pi}(\varphi_0))|_{\varphi=\varphi_0} = \nabla_{\varphi} J(\varphi)|_{\varphi=\varphi_0}.$$

Оценим полезность новой стратегии, полученной оптимизацией суррогатной функции полезности. В качестве иллюстрации свойств этого оптимизационного процесса рассмотрим смешанную стратегию, построенную скользящим средним из старой стратегии и новой, полученной при полном совершении градиентного шага:

$$\tilde{\pi}(s) = (1 - \alpha)\hat{\pi}(s; \varphi) + \alpha\hat{\pi}(s; \varphi').$$

В этом случае оказывается, что можно гарантировать следующую нижнюю границу полезности новой стратегии $\tilde{\pi}$:

$$J(\tilde{\pi}) \geq L_{\hat{\pi}(\varphi)}(\tilde{\pi}) - \frac{2\varepsilon\gamma}{(1-\gamma)^2}\alpha^2,$$

где $\varepsilon = \max_s |\mathbb{E}_{a \sim \pi(s; \varphi')}[A^{\hat{\pi}(\varphi)}(a, s)]|$.

Для оценки нижней границы полезности стохастической стратегии в общем виде справедлива следующая теорема.

Теорема 1.1.5 (О нижней границе полезности стратегии [345]). *Пусть общее статистическое расстояние (total variance) $D_{TV}^{max}(\pi_1, \pi_2) = \max_s D_{TV}^{max}(\pi_1(\cdot|s), \pi_2(\cdot|s))$; тогда*

$$J^{\hat{\pi}(\varphi')} \geq L_{\hat{\pi}(\varphi)}(\hat{\pi}(\varphi')) - \frac{4\varepsilon\gamma}{(1-\gamma)^2}(D_{TV}^{max}(\hat{\pi}(\varphi), \hat{\pi}(\varphi')))^2,$$

где $\varepsilon = \max_{s,a} |A^{\hat{\pi}(\varphi)}(a, s)|$

Известно, что общее статистическое расстояние является нижней оценкой дивергенции Кульбака-Лейблера $D_{TV}(p, q)^2 \leq D_{KL}(p, q)^2$ для логарифмических распределений p и q . Следствием теоремы 1.1.5 является следующее выражение для нижней границы через KL-дивергенцию:

$$J^{\hat{\pi}(\varphi')} \geq L_{\hat{\pi}(\varphi)}(\hat{\pi}(\varphi')) - \frac{4\varepsilon\gamma}{(1-\gamma)^2}D_{KL}^{max}(\hat{\pi}(\varphi), \hat{\pi}(\varphi')),$$

где $D_{KL}^{max}(\pi_1, \pi_2) = \max_s D_{KL}^{max}(\pi_1(\cdot|s), \pi_2(\cdot|s))$ (см. рисунок 1.6).

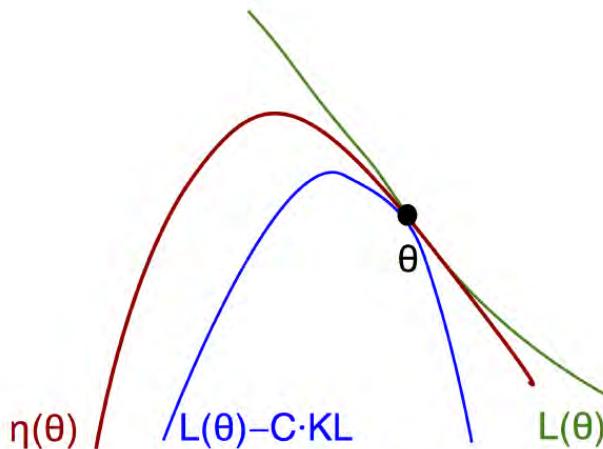


Рисунок 1.6 — Иллюстрация принципа, заложенного в теорему о нижней границе полезности стратегии. Функция $\eta(\theta)$ — истинная функция полезности, $L(\theta)$ — суррогатная функция.

Максимизация целевой функции, определяемой нижней границей оценки, гарантирует монотонное улучшение стратегии:

$$\begin{aligned} M_i(\pi) &= L_{\pi_i}(\pi) - \frac{4\epsilon\gamma}{(1-\gamma)^2}(D_{KL}^{\max}(\pi_i, \pi)) \\ J^{\pi_{i+1}} &\geq L_{\pi_i}(\pi) - \frac{4\epsilon\gamma}{(1-\gamma)^2}(D_{KL}^{\max}(\pi_i, \pi)) = M_i(\pi_{i+1}) \\ J^{\pi_i} &= M_i(\pi_i) \\ J^{\pi_{i+1}} - J^{\pi_i} &\geq M_i(\pi_{i+1}) - M_i(\pi_i) \end{aligned}$$

Другими словами, до тех пор, пока стратегия равна или улучшена по сравнению со старой стратегией в отношении к нижней границе, мы гарантируем монотонное улучшение стратегий относительно исходной функции полезности.

Перепишем в более компактном виде целевую функцию, максимизацией которой мы будем заниматься:

$$L_\varphi(\varphi') - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\varphi, \varphi') = L_\varphi(\varphi') - C \cdot D_{KL}^{\max}(\varphi, \varphi')$$

где C — штрафной коэффициент.

При практической реализации данного оптимизационного процесса с использованием коэффициента C , равного значению, рекомендуемому теорией, градиентные шаги оказываются очень маленькими, что существенно замедляет процесс обучения. Однако можно использовать так называемые *доверительные области* (trust regions) для ограничения градиентного шага. Это можно сделать, наложив ограничение на константу при KL-дивергенции между новой и старой стратегией. Получим следующую оптимизационную задачу:

$$\left. \begin{array}{l} \max_{\varphi'} L_\varphi(\varphi') \\ \text{при условии } D_{KL}^{s \sim \mu_\varphi}(\varphi, \varphi') \leq \delta \end{array} \right\} \quad (1.24)$$

Здесь используется средняя KL-дивергенция вместо максимальной (максимум требует ограничения KL во всех состояниях, что приводит к слишком большому числу ограничений).

Напомним вид суррогатной функции потерь:

$$L_\varphi(\varphi') = J(\varphi) + \sum_s \mu_{\hat{\pi}(\varphi)}(s) \sum_a \hat{\pi}(s; \varphi') A^{\hat{\pi}(\varphi)}(s, a).$$

Так как в этом выражении не известны ни точные частоты посещений состояний, ни истинное значение функции преимущества, требуется их оценка через доступную агенту в момент времени t статистику текущего набора траекторий, полученных с помощью стратегии $\hat{\pi}(s; \varphi)$.

Для получения вычислимой оценки суррогатной функции на первом шаге проведем замену частот состояний:

$$\sum_s \mu_{\hat{\pi}(\varphi)}(s)[\dots] \rightarrow \frac{1}{1-\gamma} \mathbb{E}_{s \sim \mu_\varphi} [\dots].$$

Затем заменим свертку по действиям, выполняемым по новой стратегии $\hat{\pi}(\varphi')$, на свертку по действиям, выполняемым старой стратегией $\hat{\pi}(\varphi)$, с учетом выборки по значимости:

$$\sum_a \hat{\pi}(s; \varphi') A^{\hat{\pi}(\varphi)}(s, a) \rightarrow \mathbb{E}_{a \sim \hat{\pi}(\varphi)} \left[\frac{\hat{\pi}(s; \varphi')}{\hat{\pi}(s; \varphi)} A^{\hat{\pi}(\varphi)}(s, a) \right].$$

Также на решении оптимизационной задаче не скажется замена функции преимущества $A^{\hat{\pi}(\varphi)}(s, a)$ на функцию полезности действия $Q^{\hat{\pi}(\varphi)}(s, a)$. Такая замена упрощает использование «критика» для вычисления суррогатной функции потерь.

В итоге получается следующая оптимизационная задача, в которой уже есть возможность вычислить все составляющие:

$$\max_{\varphi'} \mathbb{E}_{s \sim \mu_\varphi, a \sim \hat{\pi}(\varphi)} \left[\frac{\hat{\pi}(s; \varphi')}{\hat{\pi}(s; \varphi)} Q^{\hat{\pi}(\varphi)}(s, a) \right] \quad (1.25)$$

при условии $\mathbb{E}_{s \sim \mu_\varphi} D_{KL}(\hat{\pi}(\varphi), \hat{\pi}(\varphi')) \leq \delta$

Решение данной задачи условной оптимизации можно получить с помощью метода множителей Лагранжа:

$$\max_{\varphi'} \mathbb{E}_t \left[\frac{\hat{\pi}(s_t; \varphi')}{\hat{\pi}(s_t; \varphi)} Q^{\hat{\pi}(\varphi)}(s_t, a_t) \right] - \beta \mathbb{E}_t D_{KL}(\hat{\pi}(\varphi), \hat{\pi}(\varphi')), \quad (1.26)$$

, где вычисление матожидания идет по шагам t траекторий, набранным на текущей итерации взаимодействия агента со средой.

Для решения оптимизационной задачи 1.26 был предложен приближенный метод, который получил название *оптимизация стратегии по доверительным областям* (TRPO, Trust Region Policy Optimization) [625]. В данном методе предлагается использовать линейно-квадратичное приближение суррогатной функции в окрестности текущих параметров стратегии $\hat{\pi}(\varphi)$. Учет штрафа приводит к необходимости вычислять так называемый сопряженный градиент суррогатной функции $H^{-1}\hat{g}$, где H — гессиан усредненной KL-дивергенции (информационная матрица Фишера), а \hat{g} — стандартная оценка градиента стратегии с функцией преимущества. Итоговый вариант алгоритма TRPO выглядит следующим образом:

$$\begin{cases} \mathcal{D}_k \sim \hat{\pi}(\varphi_k) \\ \hat{A}(a_t, s_t; \theta_k), R(s_t) \leftarrow \mathcal{D}_k \\ \hat{g}_k = \frac{1}{m} \sum_{i=1}^m \sum_{t=0}^{T-1} \nabla_\varphi \log \pi(s_t; \varphi)|_{\varphi_k} \hat{A}(a_t, s_t; \theta_k) \\ \varphi_{k+1} = \varphi_k + \alpha^j \sqrt{\frac{2\delta}{(H_k^{-1}\hat{g}_k)^\top \hat{g}_k}} H_k^{-1}\hat{g}_k \\ \theta_{k+1} = \arg \min_\theta \frac{1}{mT} \sum_{i=1}^m \sum_{t=0}^{T-1} \left(\hat{V}(s_t; \theta) - R(s_t) \right), \end{cases} \quad (1.27)$$

где \mathcal{D}_k — набор траекторий, полученных текущей стратегии, m — количество этих траекторий, а $R(s_t)$ — сумма вознаграждений по траектории, начиная с состояния s_t (reward-to-go). α^j — значение градиентного шага, который выбирается минимальным из некоторого фиксированного множества значений.

Предложенный алгоритм обладает рядом существенных недостатков. Во-первых, вычисление сопряженного градиента является вычислительно затратной процедурой.

Во-вторых, данный метод показывает плохие результаты при использовании со сложными нейросетевыми аппроксиматорами, содержащими сверточные и рекуррентные слои. В связи с этим алгоритм TRPO в основном используется для задач управления с небольшими векторными пространствами состояний.

С целью упрощения методов оптимизации стратегии была предложена реализация решения оптимационной задачи 1.26 без использования методов второго порядка. Данный подход получил название *оптимизация ближайшей стратегии* (PPO, Proximal Policy Optimization) [512]. В одной из его версий используется так называемая *усеченная суррогатная функция* в качестве основного функционала качества стратегии (см. рисунок 1.7):

$$L_{\varphi_k}^{CLIP}(\varphi) = \mathbb{E}_t \left[\min \left(\frac{\pi(s_t; \varphi)}{\pi(s_t; \varphi_k)} \hat{A}_t, \text{clip}_{1-\varepsilon, 1+\varepsilon} \frac{\pi(s_t; \varphi)}{\pi(s_t; \varphi_k)} \hat{A}_t \right) \right],$$

где $\varepsilon > 0$ — небольшое значение, определяющее границы отклонения новой стратегии от текущей.

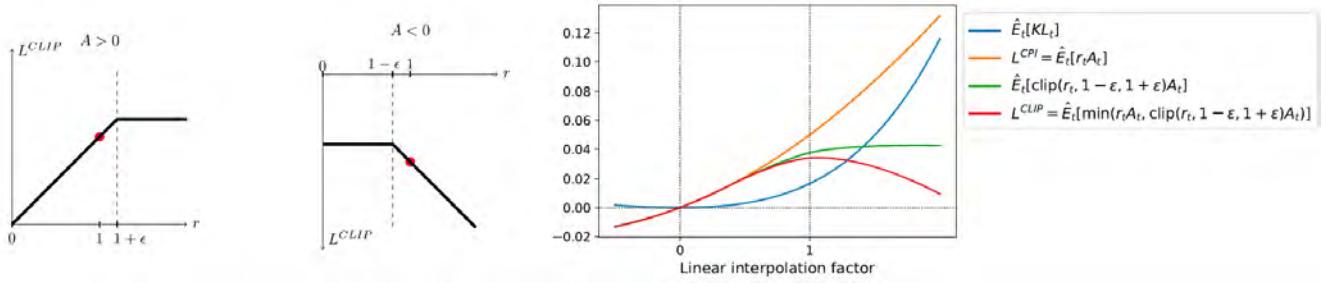


Рисунок 1.7 — Усеченная функция потерь L^{CLIP} для алгоритма РРО.

Итоговый алгоритм РРО с использованием усеченной суррогатной функции выглядит следующим образом:

$$\begin{cases} \mathcal{D}_k \sim \hat{\pi}(\varphi_k) \\ \hat{A}(a_t, s_t; \theta_k), R(s_t) \leftarrow \mathcal{D}_k \\ \varphi_{k+1} = \arg \max_{\varphi} L_{\varphi_k}^{CLIP}(\varphi) \\ \theta_{k+1} = \arg \min_{\theta} \frac{1}{mT} \sum_{i=1}^m \sum_{t=0}^{T-1} \left(\hat{V}(s_t; \theta) - R(s_t) \right). \end{cases} \quad (1.28)$$

Алгоритм РРО в настоящее время является одним из самых часто используемых методов обучения с подкреплением. Он стабильно работает с различными аппроксиматорами и применим как для дискретных, так и для непрерывных действий. Различные модификации этого метода, предложенные в последнее время (MPO [424], DD-PPO [206], Muesli [449]), улучшают его свойства сходимости и расширяют его применение на еще более сложные среды, где требуется в том числе и улучшенная стратегия исследования среды.

1.2 Планирование поведения по известной модели

1.2.1 Планирование при классических допущениях

Задача планирования представляет собой выбор и организацию действия для изменения состояний среды или, в общем случае, системы, включающей в себя как собственно среду, так и действующего агента. В связи с этим обычно для постановки задач планирования используют общую модель динамических систем в виде *системы с изменяющимися состояниями* (или *системы дискретных событий*) [207]. Формально определение такой системы следующее.

Определение 1.2.1 (Система с изменяющимися состояниями). *Система с изменяющимися состояниями — это четверка $\Sigma = \langle S, A, E, T \rangle$, где:*

- $S = \{s_1, s_2, \dots\}$ — конечное или перечислимое множество состояний,
- $A = \{a_1, a_2, \dots\}$ — конечное или перечислимое множество действий,
- $E = \{e_1, e_2, \dots\}$ — конечное или перечислимое множество событий,
- $T : S \times A \times E \rightarrow 2^S$ — функция переходов (изменений) состояний.

Такая система представима в виде направленного графа, в котором узлы соответствуют состояниям из S . Если $s' \in T(s, u)$, где $u = (a, e)$ это пара действия $a \in A$ и события $e \in E$, тогда граф содержит дугу, соединяющую s и s' и помеченную меткой u . Каждая такая дуга называется переходом состояния. Также вводится специальное нейтральное событие ε , чтобы отметить переходы, которые происходят только в связи с применением действий. Аналогично вводится нейтральное действие *noop* для отметки переходов, вызванных только событиями.

События и действия отвечают за динамику системы и отличаются тем, контролируется ли данный переход агентом-планировщиком. Действия контролируются и выполняются самим агентом. При этом, если множество $T(s, a, \varepsilon) = T(s, a)$ не пусто, то это означает, что действие a применимо в состоянии s . События соответствуют непредвиденным переходам, не контролируются агентом и являются отражением внутренней динамики среды. Если $T(s, noop, e) = T(s, e)$ не пусто, значит, событие e может произойти в состоянии e .

В общем случае предполагается, что действия и события могут происходить одновременно. В некоторых частных случаях систем с изменяющимися состояниями, например, в модели марковских игр, существует разделение множества состояний на два подмножества: в одном применимы только действия, во втором происходят только события.

В рамках данной системы Σ задачей планирования является поиск действий, применимых в соответствующих состояниях, для достижения цели начиная с некоторой исходной ситуации. Сам план — это некоторая структура, которая выдает необходимые действия. Цель может задаваться разными способами:

- В простейшем сценарии определяется целевое состояние s_g или множество целевых состояний S_g . В этом случае цель достигается некоторой последовательностью переходов состояний, завершающейся в любом целевом состоянии.
- В более общем случае цель достигается за счет удовлетворения некоторых условий, налагаемых на последовательность состояний. Например, такие условия могут заключаться в избегании некоторых состояний или, наоборот, в посещении некоторых состояний в определенный момент времени.
- Альтернативным вариантом задачи цели планирования является спецификация функции полезности состояний и оптимизации некоторой совокупной функции на множестве полезностей по последовательности состояний. Такое определение связывает задачу планирования с задачей обучения с подкреплением.

Общая схема взаимодействия агента и среды в задаче планирования выглядит следующим образом:

1. Система с изменяющимися состояниями Σ эволюционирует согласно функции переходов T в соответствии с действиями и событиями, которые она получает.
2. *Контроллер*, активная часть агента, по текущему состоянию системы генерирует действие в соответствии с некоторым планом.
3. *Планировщик*, делиберативная часть агента, по описанию системы Σ , начальной ситуации и некоторой цели, синтезирует план для контроллера для достижения этой цели.

В общем виде информация, которую получает контроллер о текущем состоянии системы, не является полной. Как и в задаче обучения с подкреплением, для моделирования частности знаний о состоянии вводится *функция наблюдений* $\Omega : S \rightarrow O$, отображающая множество состояний в некоторое дискретное множество $O = \{o_1, o_2, \dots\}$ возможных наблюдений. Входом контроллера в таком случае является наблюдение $o = \Omega(s)$, соответствующее текущему состоянию s .

В отличие от контроллера, планировщик напрямую не связан с системой переходов Σ и выполняет свою задачу по планированию автономно (*offline*). Контроллер же, напротив, напрямую взаимодействует с системой Σ и выполняет действия в реальном времени (*online*). Также необходимо отметить, что система Σ не всегда точно соответствует моделируемой реальной физической системе. Предполагается, что в некоторых пределах это несоответствие обрабатывается на уровне контроллера, но в тех случаях, когда расхождение становится слишком большим, необходимо введение обратной связи между контроллером и планировщиком. Обычно это осуществляется за счет передачи *статуса выполнения* от контроллера, а планировщик в свою очередь должен обладать функцией отслеживания плана, его обновления и полного перепланирования.

Классические методы планирования обычно используют следующие допущения:

Конечность: в системе Σ конечное множество состояний.

Полная наблюдаемость: агенту, реализующему план, доступна полная информация о текущем состоянии среды, то есть наблюдение совпадает с информационным

состоянием среды, в котором содержится все необходимое для оптимального принятия решения. Иными словами, функция наблюдений Ω является функцией тождества.

Детерминированность: действия имеют детерминированный эффект, то есть, будучи применимы в конкретном состоянии, они приводят агента в единственное новое состояние среды. Формально для каждого состояния s и для каждого события или действия u справедливо $|T(s,u)| \leq 1$.

Статичность: множество событий E пусто, то есть система не имеет своей собственной внутренней динамики и остается в том же состоянии пока контроллер не выполнит действие.

Ограниченнность цели: целями планировщика являются множества состояний и общей задачей агента является построение последовательности переходов, ведущей к одному из целевых состояний.

Последовательные планы: итоговый план является линейно упорядоченной конечной последовательностью действий.

Неявное время выполнения: действия и события не имеют длительности во времени и соответствуют мгновенным переходам состояний. В системах с изменяющимися состояниями время не представлено в явном виде.

Автономное планирование: планировщик не связан обратной связью с контроллером и не обладает информацией о каких-либо изменениях в системе Σ в процессе планирования.

В простейшем случае *ограниченной модели*, когда выполняются все перечисленные допущения, задача планирования формулируется следующим образом:

Определение 1.2.2 (Классическая задача планирования). *При заданной системе $\Sigma = \langle S, A, T \rangle$, начальном состоянии s_0 и подмножестве целевых состояний S_g необходимо найти последовательность действий $\langle a_1, a_2, \dots, a_k \rangle$, соответствующих последовательности переходов состояний $\langle s_0, s_1, \dots, s_k \rangle$ такую, что $\forall i \in \{1, k\} s_i \in T(s_{i-1}, a_i) \text{ и } s_k \in S_g$.*

Данная классическая задача сводится к задаче поиска пути в графе Σ и основной сложностью здесь является поиск такого представления состояний, в соответствии с которым полезное подмножество состояний будет компактным и будет давать возможность осуществить эффективный поиск (см. рисунок 1.8).

1.2.2 Представление состояний в задаче планирования

В большинстве задач планирования, даже с учетом классическим допущений, пространство состояний достаточно велико. Обычно в таком случае состояния не описываются и не перечисляются напрямую. Вместо этого используется так называемое *представление состояний*. Правильный выбор представления позволяет формировать явное описание состояния и функцию переходов в процессе планирования на лету.

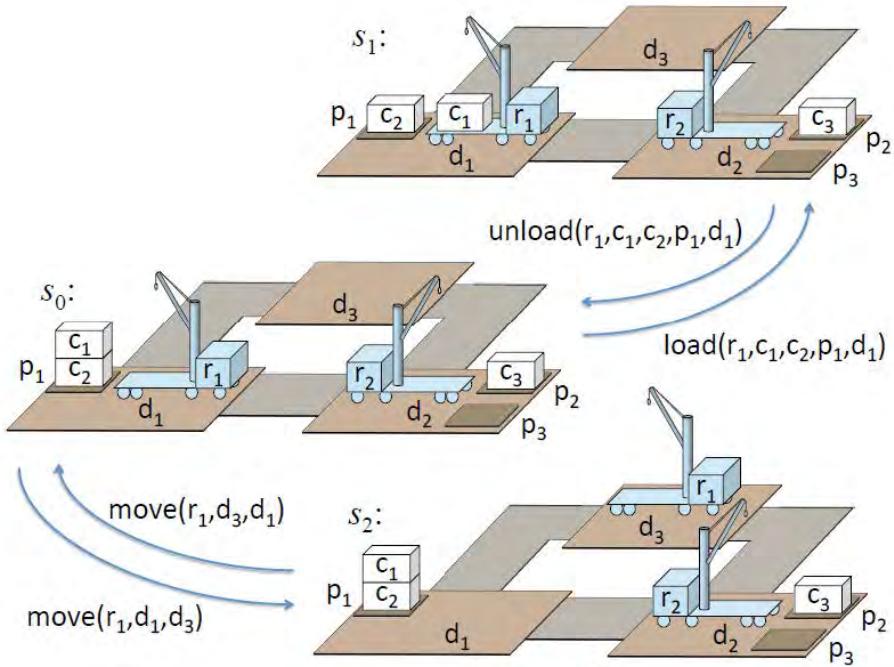


Рисунок 1.8 — Пример переходов между состояниями при классическом планировании в задаче перемещения грузов погрузчиками с действиями выгрузить груз (unload), погрузить груз (load), переехать на новую площадку (move).

В классическом планировании традиционно используются следующие подходы к представлению состояний:

Теоретико-множественное представление: каждое состояния среды может быть представлено в виде множества пропозиций [676], а каждое действие является синтаксическим выражением, в котором применимость выражается через принадлежность пропозиционного символа текущему состоянию, а результат действия добавляет или удаляет новые пропозиционные символы;

Классическое представление: аналогичен теоретико-множественному кроме того, что проведена замена пропозиций на выражения логики предикатов первого порядка, является наиболее популярным вариантом представления в классическом планировании;

Представление с переменными состояния: каждое состояние описывается множеством значений переменных или признаков, а действие является частичной функцией, отображающей это множество в некоторое другое множество признаков.

С введением понятия представления состояний появляется возможность определить термин домена планирования и уточнить задачу планирования и ее решение.

Теоретико-множественное представление

Определение 1.2.3 (Теоретико-множественный домен планирования). Пусть $L = \{p_1, \dots, p_n\}$ — конечное множество пропозиционных символов, тогда теоретико-множественным доменом планирования на L является следующая ограниченная система с изменяющимися состояниями $\Sigma = \langle S, A, T \rangle$:

- $S \subseteq 2^L$, т.е. каждое состояние s является подмножеством L . Если $p \in s$, то пропозиция p выполняется в состоянии среды, представленном s , а если $p \notin s$, то, наоборот, p не выполняется в состоянии, представленном s .
- Каждое действие $a = \langle \text{precond}(a), \text{effects}^-(a), \text{effects}^+(a) \rangle$ является тройкой подмножеств из L : множество $\text{precond}(a)$ называется предусловиями действия a , а множества $\text{effects}^-(a)$ и $\text{effects}^+(a)$ называются эффектами действия, при этом $\text{effects}^-(a) \cap \text{effects}^+(a) = \emptyset$. Действие a применимо к состоянию s , если $\text{precond}(a) \subseteq s$.
- S обладает тем свойством, что если $s \in S$, то для каждого применимого к s действия a выполняется $(s - \text{effects}^-(a)) \cup \text{effects}^+(a) \in S$. Это означает, что любое применение действия вызывает переход к некоторому новому корректному состоянию, что позволяет при заданном A определить S через небольшое множество начальных состояний.
- Функция переходов T определяется как $T(s, a) = (s - \text{effects}^-(a)) \cup \text{effects}^+(a)$, если $a \in A$ применимо к $s \in S$, иначе $T(s, a)$ не определено.

Определение 1.2.4 (Теоретико-множественная задача планирования). Теоретико-множественной задачей планирования называется тройка $\mathcal{P} = \langle \Sigma, s_0, g \rangle$, где:

- $s_0 \in S$ — начальное состояние,
- $g \subseteq L$ — множество целевых пропозиций, задающих требования, которые должны выполняться в одном из целевых состояний: $S_g = \{s \in |g \subseteq s\}$.

При использовании теоретико-множественного представления план π уточняется как любая последовательность действий $\langle a_1, \dots, a_k \rangle$, $k \geq 0$. Длина плана $|\pi| = k$ определяется как количество действий. Если заданы два плана $\pi_1 = \langle a_1, \dots, a_k \rangle$, $\pi_2 = \langle a'_1, \dots, a'_j \rangle$, то операция конкатенации планов определяется как

$$\pi_1 \cdot \pi_2 = \langle a_1, \dots, a_k, a'_1, \dots, a'_j \rangle.$$

Если задан некоторый план $\pi = \langle a_1, \dots, a_k \rangle$, то можно ввести траекторию системы, задаваемую функцией переходов T с учетом последовательного применения действий из плана:

$$T(s, \pi) = \begin{cases} s, & \text{если } k = 0, \\ T(T(s, a_1), \langle a_2, \dots, a_k \rangle), & \text{если } k > 0 \text{ и } a_1 \text{ применимо к } s, \\ \text{не определено} & \text{иначе.} \end{cases}$$

Определение 1.2.5 (Решение теоретико-множественной задачи планирования). Пусть $\mathcal{P} = \langle \Sigma, s_0, g \rangle$ — задача планирования, тогда план π является решением задачи \mathcal{P} , если $g \subseteq T(s_0, \pi)$. Решение является избыточным, если существует подходящая подпоследовательность в π , которая также является решением \mathcal{P} . π является минимальным, если ни одно другое решение \mathcal{P} не содержит меньше действий, чем π .

Классическое представление

В классическом представлении состояний для задачи планирования используется подмножество языка предикатов первого порядка [671; 676], в котором отсутствуют функциональные символы и добавляются некоторые специфичные символы и выражения. Состояния в этом случае представляются как множество логических атомов (атомарных формул без кванторов и логических связок), принимающие значение 0 (ложь) или 1 (истина) в рамках некоторой интерпретации. Действия представляются в виде операторов планирования, меняющих истинностные значения этих атомов.

Пусть \mathcal{L} — язык предикатов первого порядка с конечными множествами предикатных и константных символов и без функциональных символов. Состояние будет описываться как множество атомов языка \mathcal{L} без переменных или фактов. В отсутствие функциональных символов множество возможных состояний конечно. Если g является множеством литералов (атомов или отрицаний атомов), то $s \models g$ будет означать, что s удовлетворяет g , когда имеется некоторая подстановка σ такая, что каждый литерал из $\sigma(g)$ без отрицания принадлежит s и ни один литерал из $\sigma(g)$ с отрицанием не принадлежит s . Необходимо отметить, что в задачах планирования с использованием классического представления подразумевается выполнение *допущения замкнутого мира*, то есть выполнения условия, при котором отсутствие атома в определении состояния означает то, что он не выполняется в этом состоянии.

Как правило, множество предикатов делится на две группы. Первая группа — это *переменные отношения*, являющиеся функциями на множестве состояний. Истинностные значения таких отношений могут меняться от состояния к состоянию. Вторая группа — это *постоянные отношения* истинностные значения которых остаются константными и не меняются от одного состояния к другому.

Определение 1.2.6 (Оператор планирования). Оператором планирования в классическом планировании будет называться тройка

$$o = \langle name(o), precond(o), effects(o) \rangle,$$

где:

- $name(o)$ — имя оператора, синтаксическое выражение вида $n(x_1, \dots, x_k)$, в котором n — уникальный в \mathcal{L} символ оператора, x_1, \dots, x_k — символы переменных, встречающихся в o ;

- $\text{precond}(o)$ и $\text{effects}(o)$ – предусловия и эффекты оператора o соответственно, являются обобщениями аналогичных понятий в теоретико-множественном представлении, то есть множествами литералов.

Действием a в классическом планировании называется любой экземпляр оператора планирования без переменных, то есть оператор с выполненной подстановкой переменных. Если состояние s такое, что позитивные предусловия (то есть литералы без отрицания) $\text{precond}^+(a) \subseteq s$ и негативные предусловия (то есть литералы с отрицанием) $\text{precond}^-(a) \cap s = \emptyset$, то действие a применимо к s , а результатом применения a к s является

$$T(s,a) = (s - \text{effects}^-(a)) \cup \text{effects}^+(a).$$

Определение 1.2.7 (Классический домен планирования). Пусть \mathcal{L} – язык предикатов первого порядка с конечным множеством предикатных и константных символов, а F – множество всех фактов в \mathcal{L} . Тогда классическим доменом планирования в \mathcal{L} будет являться ограниченная система с переменными состояниями $\Sigma = \langle S, A, T \rangle$ такая, что:

- $S \subseteq 2^F$;
- A – множество всех экземпляров операторов без переменных из множества всех операторов O ;
- $T(s,a) = (s - \text{effects}^-(a)) \cup \text{effects}^+(a)$, если $a \in A$ применимо к $s \in S$, иначе $T(s,a)$ не определено;
- S замкнуто относительно T , то есть если $s \in S$, тогда для любого a , примененного к s , $T(s,a) \in S$.

Определение 1.2.8 (Классическая задача планирования). Классической задачей планирования называется тройка $\mathcal{P} = \langle \Sigma, s_0, g \rangle$, где:

- s_0 – начальное состояние из множества S ,
- g – цель, некоторое множество фактов и их отрицаний,
- $S_g = \{s \in S | s \models g\}$.

Постановкой задачи планирования \mathcal{P} будет называться тройка $P = \langle O, s_0, g \rangle$. Несколько задач планирования могут иметь одну и ту же постановку. В этом случае у них одно и тоже множество решений. Постановка задачи является описанием синтаксиса, используемого при поиске решения задачи планирования. Данное разделение аналогично разделению логической теории и ее модели [669] (см. рисунок 1.9). Если I – это интерпретирующая функция такая, что $\forall t \in P I(t) = s \in \Sigma$ и $\forall o \in P I(o) = a \in \Sigma$, то $\langle \Sigma, I \rangle$ называется моделью P , если $\forall t \in P.o \in P$ справедливо:

$$T(I(t), I(o)) = I((s - \text{effects}^-(o)) \cup \text{effects}^+(o)).$$

Определение (минимального) плана в классическом случае аналогично теоретико-множественному случаю. План π является решением задачи планирования \mathcal{P} , когда $T(s_0, \pi) \models g$.

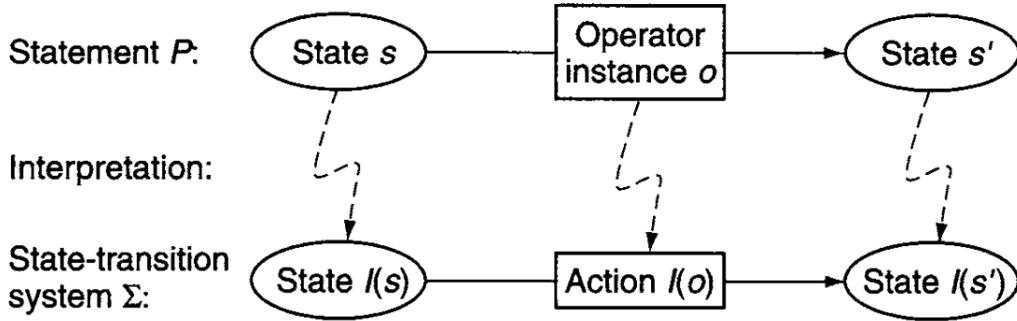


Рисунок 1.9 — Интерпретация домена планирования как системы с переменными состояниями.

Расширения классического представления в задаче планирования (например, введение классов констант или условных операторов планирования) являются наиболее часто используемыми языками представления задач. На их основе строится и самый распространенный язык описания доменов и задач планирования — PDDL [266; 281].

Представление с переменными состояния

Третий из перечисленных выше вариантов представления состояний в задаче планирования, с переменными состояний, эквивалентен в смысле выразительной способности двум предыдущим вариантам. Одной из его основных особенностей является использование функций, зависящих от состояния, вместо переменных отношений в классическом случае.

Пусть D — конечное множество всех константных символов в домене планирования, которое разделяется на классы констант, обозначающих различные типы объектов среды. Определим объектную переменную как один из символов переменных, сопоставленный некоторому подмножеству из множества констант. Каждая объектная переменная v соответствует интервалу D^v , являющемуся объединением одной или нескольких классов констант.

Тогда k -арной переменной состояния будет являться выражение вида $x(v_1, \dots, v_k)$, где x — символ переменной состояния, а каждая v_i является либо объектным символом, либо объектной переменной. Переменная состояния обозначает элемент функции переменной состояния:

$$x : D_1^x \times \dots \times D_k^x \times S \rightarrow D_{k+1}^x,$$

где каждый $D_i^x \subseteq D$ является объединением одного или более классов.

Переменная состояния $x(c_1, \dots, c_k)$ определена, когда каждая c_i является константой в D_i^x . Наоборот, переменная $x(v_1, \dots, v_k)$ не определена, если одна или более v_1, \dots, v_k являются объектными переменными. Для k -местного фиксированного отношения $r(v_1, \dots, v_k)$ существуют множества D_1^r, \dots, D_k^r такие, что каждое D_i^r является объединением одного или более классов и $r \subseteq D_1^r \times \dots \times D_k^r$.

Определение 1.2.9 (Оператор планирования с переменными состояний). *Оператором планирования с переменными состояний будет называться тройка*

$$o = \langle name(o), precond(o), effects(o) \rangle,$$

где:

- $name(o)$ — имя оператора, синтаксическое выражение вида $n(u_1, \dots, u_k)$, в котором n — уникальный символ оператора, u_1, \dots, u_k — символы объектных переменных, встречающихся в o ;
- $precond(o)$ — множество выражений на переменных состояний и отношениях;
- $effects(o)$ — множество подстановок значений переменных состояния в виде $x(t_1, \dots, t_k) \leftarrow t_{k+1}$, где каждая t_i является термом соответствующего интервала.

Действием a является определенный оператор o , содержащий фиксированные отношения в предусловиях $precond(o)$, то есть каждая объектная переменная в o заменена на константу соответствующего класса так, что эти константы удовлетворяют фиксированным отношениям в $precond(o)$. Действие a применимо к состоянию s , если значения переменных состояния s удовлетворяют условиям в $precond(a)$. Результатом применения действия является обновление значений этих переменных состояния в соответствии с подстановками из $effects(a)$. Цель задается определением значений одной или более переменных состояния.

Определение 1.2.10 (Домен планирования с переменными состояния). *Пусть \mathcal{L} — язык планирования в представлении переменных состояния, заданный конечным множеством переменных состояний X и конечным множеством фиксированных отношений R . Тогда доменом планирования в \mathcal{L} будет являться ограниченная система с переменными состояниями $\Sigma = \langle S, A, T \rangle$ такая, что:*

- $S \subseteq \prod_{x \in X} D_x$, где D_x — интервал значений переменной состояния x ; состояние s обозначается как $s = \{(x = c) | x \in X\}$, где $c \in D_x$;
- A — все определенные экземпляры операторов из O , удовлетворяющие отношениям из R ; действие a применимо к состоянию s тогда и только тогда, когда выражение $(x = c)$ из $precond(a)$ имеется также в s ;
- $T(s, a) = \{(x = c) | x \in X\}$, где c определено через подстановку $x \leftarrow c$ в $effects(a)$, если есть такая подстановка, иначе $(x = c) \in s$;
- S замкнуто относительно T , то есть если $s \in S$, тогда для каждого действия a , применимого к s , $T(s, a) \in S$.

Определение 1.2.11 (Задача планирования с переменными состояния). *Задачей планирования в представлении с переменными состояния называется тройка $\mathcal{P} = \langle \Sigma, s_0, g \rangle$, где s_0 — начальное состояние из S , а цель g задается множеством выражений на переменных состояния из X .*

Постановкой задачи планирования в представлении с переменным состояния является четверка $P = \langle O, R, s_0, g \rangle$, где O — множество операторов, R — множество фиксированных отношений, s_0 — начальное состояние, а g — цель.

Как и в классическом случае, планом для задачи является последовательность действий $\pi = \langle a_1, \dots, a_k \rangle$ такая, что a_1 применимо к s_0 , то есть условия и отношения в $\text{precond}(a)$ удовлетворяются в состоянии s_0 , a_2 применимо к $T(s_0, a_1), \dots$, а условия цели g удовлетворяются в $T(s_{k-1}, a_k)$.

Алгоритмы поиска

Простейшими классическими алгоритмами планирования являются алгоритмы поиска в пространстве состояний. В этих подходах пространство поиска представляет собой подмножество пространства состояний: каждый узел соответствует состоянию среды, а каждая дуга — переходу между состояниями. Текущий план представим в виде текущего пути в пространстве поиска.

Пример такого алгоритма, простейший прямой поиск FORWARDSEARCH(O, s_0, g), выглядит следующим образом:

$$\begin{aligned} s_1 &\leftarrow s_0, \pi_1 \leftarrow \emptyset, \\ \begin{cases} a_k \sim \{a_i | a_i = a(o) \wedge o \in O \wedge s_k \models \text{precond}(a_i)\}, \\ s_{k+1} = T(s_k, a_k), \\ \pi_{k+1} = \pi_k \cdot \langle a_k \rangle. \end{cases} \end{aligned} \quad (1.29)$$

Здесь $a(o)$ означает определенный экземпляр оператора o .

Теорема 1.2.1 (Корректность алгоритма FORWARDSEARCH [282]). *Алгоритм поиска FORWARDSEARCH (O, s_0, g), определенный в 1.29, корректен, то есть любой план, возвращаемый этим алгоритмом, является решением задачи планирования $\langle O, s_0, g \rangle$.*

Более продвинутым алгоритмом поиска в пространстве состояний является метод STRIPS(O, s_k, g), реализующий идею поиска от целевого состояния:

$$\begin{aligned} s_1 &\leftarrow s_0, \pi_1 \leftarrow \emptyset, \\ \begin{cases} a_k \sim \{a_i | a_i = a(o) \wedge o \in O \wedge s_k \in g\}, \\ \pi' = \text{STRIPS}(O, s_k, \text{precond}(a_k)) s_{k+1} = T(s_k, \pi'), \\ s_{k+1} = T(s_{k+1}, a_k), \\ \pi = \pi' \cdot \langle a_k \rangle. \end{cases} \end{aligned} \quad (1.30)$$

STRIPS представляет собой базовый алгоритм планирования в пространстве состояний. Более продвинутые подходы используют более сложные представления состояний, а также парадигму планирования в пространстве планов (планировщик GraphPlan [157]).

Несмотря на то что существует большое количество продвинутых планировщиков, они подвержены одной и той же проблеме комбинаторной сложности процедура поиска

в больших пространствах состояний. Для сокращения перебора была предложена особая техника, используемая в планировщиках, — использование *эвристики*. Что планировщики в пространстве состояний, что планировщики в пространстве планов могут быть описаны в виде следующей обобщенной рекурсивной процедуры ABSTRACTSEARCH(u), принимающей на вход некоторый узел u и пространства поиска:

$$\begin{cases} u = \text{Refine}(u), \\ B = \text{Branch}(u), \\ C = \text{Prune}(B), \\ v \sim C, \\ \text{ABSTRACTSEARCH}(v). \end{cases} \quad (1.31)$$

Здесь предполагается, что u представляет набор планов (решений задачи) Π_u . Например, в пространстве состояний это просто последовательность действий, а в пространстве планов это множество действий с каузальными связями и другими ограничениями на последовательность их применения. В ABSTRACTSEARCH используются следующие функции:

Refine означает модификацию множества действий и ограничений без изменения множества решений задачи Π_u .

Branch генерирует одного или нескольких потомков узла u , каждый v из них представляет некоторое подмножество решений $\Pi_v \subseteq \Pi_u$.

Prune удаление среди сгенерированных кандидатов на рассмотрение наименее перспективных с точки зрения итогового решения.

Основным шагом в этом алгоритме на данный момент является недетерминированный выбор следующего потомка для рассмотрения в новом цикле рекурсии ($v \sim C$), который в стандартных алгоритмах поиска реализуется через поиск вглубь. Именно при организации данного шага, а также и на других этапах этого обобщенного алгоритма, и используются различные эвристические техники для сокращения перебора и организации направленного поиска. Наиболее популярны доменно независимые эвристики, к которым относятся такие алгоритмы как A* [311] и Theta* [617], которые в том числе используются в данной работе.

1.2.3 Планирование в условиях неопределённости

Перечисленные в предыдущем пункте допущения, в которых работают методы классического планирования, существенно сужают область их применения. В условиях неопределенности наиболее критичными представляются ограничения на полную детерминированность, полную наблюдаемость и неполноту задания цели. Именно снятие этих ограничений и явное рассмотрение возникающих вариантов неопределенности и являются целью создания продвинутых методов планирования.

Снятие первого ограничения ведет к учету недетерминированности совершения действий и возможности работать с неидеальными моделями среды и всей системы. В этом случае план может реализовываться в виде множества различных путей выполнения. Простейшим вариантом здесь является назначения вероятностей возможным исходам действий.

Снятие второго ограничения приводит к необходимости работы с различными состояниями системы, которые не отличимы для контроллера. В некоторых случаях для получения необходимой информации необходимо выполнять так называемые сенсорные действия или действия по запросу к внешнему источнику данных. В этом случае необходимо учитывать, что наблюдению соответствует не одно информационное состояние, а несколько. Пространство поиска в этом случае строится не на множестве состояний, а на булеване состояний.

Наконец, в связи с недетерминированным выполнением действий необходимо расширять и определение цели. Расширенное определение цели в данном случае может включать введение функции полезности, и задачей планирования является поиск плана, максимизирующего такую функцию полезности. Другой вариант заключается в описании цели с помощью языка темпоральной логики.

Стандартным подходом для формализации планирования в условиях неопределенности является введение марковского процесса принятия решений по тому же пути, что проходил при постановке задачи обучения с подкреплением в Главе 1. Домен планирования в этом случае моделируется в виде стохастической системы с недетерминированными переходами между состояниями. Цели представляются в виде функции полезности, определяющей предпочтение на множестве состояний при выполнении плана. Сам план в таком случае кодируется функцией стратегии, определяющей планируемое действие для текущего состояния. Задача планирования формулируется в такой же постановке, что и задача обучения с подкреплением, то есть в виде оптимизационной задачи максимизации функции полезности. Наконец, частичная наблюдаемость моделируется наблюдениями (*доверительными состояниями*), задающими распределение на множестве состояний.

В случае полной наблюдаемости домен планирования определяется через введение стохастической системы:

Определение 1.2.12 (Стохастическая система). *Стохастической системой называется недетерминированная система с изменяющимися состояниями и с распределением вероятности на множестве вариантов перехода из состояния $\Sigma = \langle S, A, T \rangle$. Здесь:*

- S – конечное множество состояний,
- A – конечное множество действий,
- $T(s'|s,a)$, где $a \in A$, s и s' из множества S , а T – это собственно распределение вероятностей на множестве состояний. Для каждого $s \in S$, если существует $a \in A$ и $s' \in S$ такие, что $T(s'|s,a) \neq 0$, тогда справедливо $\sum_{s' \in S} T(s',s,a) = 1$.

Значение $T(s'|s,a)$ характеризует вероятность того, что, будучи применимым к состоянию s , действие a приведет к состоянию s' . Множество $A(s) = \{a \in A | \exists s' \in S, T(s'|s,a) \neq 0\}$ называется множеством *выполнимых действий* в s .

План, как и обычно, определяет действия, которые контроллер должен выполнить в данном состоянии. Как это принято и в обучении с подкреплением, план может быть представлен в виде стратегии π — всюду определенной функции, отображающей состояния в действия: $\pi : S \rightarrow A$. Данная функция генерируется планировщиком и подается на вход контроллеру, выполняющему действия последовательно, наблюдая следующее состояние системы. Выполнение стратегии приводит не к одному линейному плану, а к бесконечным последовательностям действий, представимых в виде марковских цепей, где вероятность следующего состояния зависит только от предыдущего состояния в цепи.

В отличие от классического планирования, цели в МПР представлены в виде *функций полезности*. При этом в планировании принято разделять затраты и вознаграждения, получаемые агентом. Иначе говоря, для стохастической системы Σ вводят две функции: затрат $C : S \times A \rightarrow \mathbb{R}$ и вознаграждений $R : S \rightarrow \mathbb{R}$. Полезность V в состоянии s при совершении действия a , в свою очередь, определяют как разность $V(s,a) = R(s) - C(s,a)$. Для стратегии полезность записывается как $V(s|\pi) = R(s) - C(s,\pi(s))$. Аналогично можно определить полезность последовательности состояния, или *истории*, $h = \langle s_0, s_1, \dots \rangle$, полученной с помощью стратегии π :

$$V(h|\pi) = \sum_{i \geq 0} (R(s_i) - C(s_i, \pi(s_i))).$$

Выражение для полезности, представленное выше, не обязательно сходится к конечному значению, что неудобно из-за невозможности сравнивать различные истории и, соответственно, стратегии. Именно в связи с этим вводится дисконтирующий фактор $0 < \gamma < 1$, как и в постановке задачи обучения с подкреплением. Это приводит к тому, что накопленные затраты и вознаграждения на ранних стадиях больше, чем накопленные на более поздних:

$$V(h|\pi) = \sum_{i \geq 0} \gamma^i (R(s_i) - C(s_i, \pi(s_i))).$$

Аналогично тому, как вводилась полезность стратегии в градиентных методах (см. раздел 1.1), в задаче планирования полезность стратегии является матожиданием по всем возможным историям, получаемым по этой стратегии:

$$J(\pi) = \sum_{h \in H} T(h|\pi) V(h|\pi) = \sum_{h \in H} \prod_{i \geq 0} T(s_{i+1}, s_i, \pi(s_i)) V(h|\pi),$$

где H — множество всех возможных историй для стохастической системы Σ .

Естественным образом определяется и *оптимальная стратегия* π^* в системе Σ , обладающая максимальным значением полезности $J(\pi^*) \geq J(\pi)$ среди всех возможных в Σ стратегий π . Задача планирования в таком случае представляется как задача оптимизации:

по имеющимся стохастической системе Σ и функции полезности решением является оптимальная в Σ стратегия.

В представленной выше формулировке задача планирования совпадает с задачей обучения с подкреплением, когда известна функция переходов $T(s', s, a)$. Алгоритмы планирования в этой постановке являются частными случаями обобщенной итеративной оценки стратегии 1.7 в разделе 1.1: алгоритмы оценки стратегии и полезности.

Пусть функция вознаграждения нулевая и имеются только ненулевые затраты. В таком случае матожиданием затрат будет выражаться как $V(s) = \mathbb{E}_a C(s, a) = \sum_a \pi(a|s)C(s, a)$, а функция полезности действия — как

$$Q(s, a) = C(s, a) + \gamma \sum_{s' \in S} T(s', s, a)V(s').$$

Уравнение Беллмана для оптимальной стратегии запишется следующим образом:

$$V^{\pi^*}(s) = \min_a \left(C(s, a) + \gamma \sum_{s' \in S} T(s', s, a)V^{\pi^*}(s') \right). \quad (1.32)$$

Первый вариант решения этого уравнения представляет собой алгоритм *итерации по стратегии* при некоторой начальной стратегии π_0 представляет собой решения системы из $|S|$ уравнений относительно V

$$V(s) = \min_a \left(C(s, a) + \gamma \sum_{s' \in S} T(s', s, a)V(s') \right).$$

Алгоритм выглядит следующим образом:

$$\begin{cases} V^{\pi_k}(s) \leftarrow \text{solve} \left(V^{\pi_k}(s) = C(s, a) + \gamma \sum_{s' \in S} T(s', s, \pi_k(s))V^{\pi_k}(s') \right), \\ \pi_{k+1}(s) = a, \text{ если } V^{\pi_k}(s) > C(s, a) + \gamma \sum_{s' \in S} T(s', s, a)V^{\pi_k}(s'). \end{cases} \quad (1.33)$$

На первой фазе определяется полезность, то есть оценивается текущая стратегия. А на второй фазе происходит улучшение стратегии за счет выбора действия, приводящего к меньшим затратам. Алгоритм завершает свою работу при отсутствии альтернативных действий, улучшающих стратегию ($\pi_k = \pi_{k+1}$), и сходится к оптимальному значению.

Второй вариант решения 1.32 называется итерацией по полезности и некоторому начальному значению $V_0(s)$, итеративно вычисляет следующее значение полезности в стиле динамического программирования:

$$\begin{cases} Q(s, a) \leftarrow C(s, a) + \gamma \sum_{s' \in S} T(s', s, a)V_k(s'), \\ V_{k+1} \leftarrow \min_{a \in A} Q(s, a), \\ \pi_{k+1}(s) = \arg \min_{a \in A} Q(s, a). \end{cases} \quad (1.34)$$

Несмотря на то что есть теоретические гарантии получения оптимальной стратегии за конечное множество шагов, обычно выполнение алгоритма 1.34 завершается при достижении порога изменения полезности $\max_{s \in S} |V_{k+1}(s) - V_k(s)| < \varepsilon$. Получающаяся стратегия будет, соответственно, ε -оптимальной.

Два алгоритма планирования отличаются тем, как происходит суммирование полезностей для подсчета нового значения. Итерация по стратегии более вычислительная затратная, но быстрее сходится, хотя сложность обоих алгоритмов полиномиальна относительно мощности множеств S и A . Для высокоразмерных пространств состояний данные подходы оказываются практически не применимы.

Частичная наблюдаемость

Помимо снятия ограничений в детерминированности среды и в описании целевых состояний, важным ограничением на пути к построению полноценных методов учета неопределенности в планировании является допущение полной наблюдаемости системы. Как уже было сказано ранее, в том числе и в разделе 1.1, посвященном обучению с подкреплением, данное ограничение снимается путем введения понятия наблюдения, представляющей только часть информации о состоянии, доступной агенту. В терминах МППР наблюдения соответствуют распределению вероятностей на множестве состояний системы.

Определение 1.2.13 (Частично-наблюдаемая стохастическая система). *Частично-наблюдаемой стохастической системой будем называть следующую конструкцию:*

- стохастическая система $\Sigma = \langle S, A, T \rangle$, где S, A, P определены так же, как в определении 1.2.12,
- конечное множество наблюдений O с определенными вероятностями $P(o|s, a)$ для каждого $s \in S, a \in A$ и $o \in O$, выражаяющих вероятность наблюдения o в состоянии s после выполнении действия a с условием $\sum_{o \in O} P(o|s, a) = 1$.

Основной особенностью постановки с частичной наблюдаемостью является наличие в системе Σ неразличимых состояний s и s' , то есть таких, что $\forall o \in O \forall a \in A P(o|s, a) = P(o|s', a)$. Также может оказаться, что один и те же состояния могут соответствовать различным наблюдениям $P(o|s, a) \neq P(o|s, a')$.

В частично-наблюдаемом марковском процессе принятия решений контроллер имеет доступ только к распределению вероятностей на множестве наблюдений, а не напрямую к состоянию системы. Данное распределение называется *доверительным состоянием* (belief state). Пусть $b \in \Delta(S)$ — доверительное состояние, тогда $b(s)$ будет обозначать вероятность, назначенную состоянию s доверительным состоянием b при этом $0 \leq b(s) \leq 1$ и $\sum_{s \in S} b(s) = 1$.

При заданном доверительном состоянии b результатом выполнения действия будет новое доверительное состояние b' . Для каждого $s \in S$ доверительное состояние, получившееся после применения действия a в доверительном состоянии b , считается следующим образом:

$$b'(s|a) = \sum_{s' \in S} P(s'|s, a) b(s'). \quad (1.35)$$

Вероятность получить наблюдение $o \in O$ после выполнения действия a считается следующим образом:

$$b(o|a) = \sum_{s \in S} P(o|s,a)b(s). \quad (1.36)$$

Наконец, вероятность нахождения в состоянии s после выполнения действия a в доверительном состоянии b с наблюдением o считается следующим образом:

$$b(s|o,a) = \frac{P(o|s,a)b(s|a)}{b(o|a)}. \quad (1.37)$$

Стратегией в частично-наблюдаемом МППР является функция, отображающая доверительные состояния в действиях: $\pi : \Delta(S) \rightarrow A$. В то время как множество состояний S конечно, множество $\Delta(S)$ бесконечно и непрерывно. Задачей планирования в данном случае будет являться задача поиска оптимальной стратегии, которая может быть сформулирована как задача планирования в МППР с бесконечным пространством доверительных состояний. Уравнение Беллмана в такой постановке запишется как

$$V(b) = \min_{a \in A} \sum_{s \in S} C(s,a)b(s) + \gamma \sum_{o \in O} b(o|a)V(b(o|a)). \quad (1.38)$$

Так как пространство доверительных состояний бесконечно, то решение данной оптимизационной задачи может быть получено только с использованием аппроксимации функции полезности или самой стратегии.

1.3 Интеграция планирования и обучения

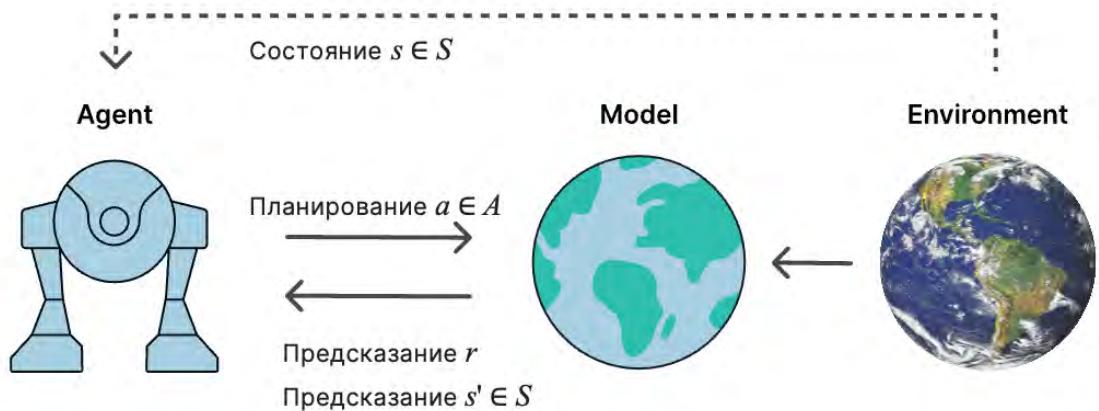


Рисунок 1.10 — Обобщенная схема работы методы обучения с подкреплением на основе модели с этапом планирования по модели.

Как задача планирования поведения, так и задача обучения с подкреплением относятся к классу задач последовательного принятия решений. Одним из вариантов интегрированной

постановки задачи является так называемое *машинное обучение с подкреплением на основе модели* (ОПМ, MBRL, model-based reinforcement learning). В ОПМ выделяется две основных подзадачи: автоматическое формирование модели динамики среды в условиях недостаточной информации, неопределённости поведения среды, стохастичности и т.д., а также собственно определение условий запуска процедуры планирования по этой модели (см. рисунок 1.10). Множество примеров демонстрирует, что правильная интеграция этапов обучения и планирования в одном цикле взаимодействия агента со средой приводит к повышению эффективности формирования стратегии поведения агента. Исследования в области ОПМ являются наиболее близкими к методам, развивающимся в настоящей работе, поэтому данный раздел будет посвящён их аналитическому обзору и сопоставлению основных преимуществ и недостатков.

1.3.1 Обучение модели среды

Класс алгоритмов обучения с моделью

Ключевым отличием обучения с подкреплением и планирования является доступность модели динамики среды. В процессе планирования известна точная модель (функции T и R), в то время как в процессе обучения они не известны. При этом под моделью среды в планировании подразумевается доступ к *обращаемой* модели, которая может вызываться в любой точке пространства состояний и действий. В обучении с подкреплением эпизоды *необратимы*, то есть нет возможности вернуться в предыдущее состояние после совершения действия. В интеграционном подходе, обучении с подкреплением на основе модели, предполагается, что обучение начинается в неизвестной необратимой среде, но получаемые данные используются для обучения модели динамики. Данная модель формируется обратимой, что позволяет использовать ее для различных алгоритмов планирования.

Ряд моделей, которые используют известную модель, но обучают функции полезности и стратегии, также относятся к классу ОПМ-алгоритмов. Такая комбинация обучения и планирования в основном применима в полностью наблюдаемой среде и обычно используется для небольшого пространства состояний [422]. В общем, класс ОПМ-алгоритмов описывается следующим образом:

Определение 1.3.1 (Класс ОПМ-алгоритмов). *К классу алгоритмов обучения с подкреплением на основе модели относится любой метод, решающий задачу МПР, использующий модель среды (известную или обучаемую) и формирующий функции стратегии и/или полезности для выполнения действий в среде.*

Таким образом, следуя определению 1.3.1, очевидно, что к ОПМ-алгоритмам не относятся следующие подходы:

- методы планирования по обучаемой модели динамики среды, не использующие обучения для формирования функций стратегии или полезности;
- планирование действий, в котором этапы обучения модели, собственно планирования и аппроксимации стратегии и/или полезности объединены в цикл оптимизации по типу «черного ящика»;
- методы, использующие базу прецедентов или эпизодическую память, так как они только воспроизводят наблюдаемые данные, но не формируют их обобщение [518].

Обучение одношаговых моделей

Обучение модели естественным образом формализуется как задача обучения с учителем [341], к которой применимо множество известных алгоритмов. Рассмотрим различные типы моделей, для которых различаются сами задачи обучения. Пусть имеется некоторый прецедент $\{s_t, a_t, r_t, s_{t+1}\}$ или пакет таких прецедентов.

Модель вознаграждения: $(s_t, a_t) \rightarrow$. Задача обучения формулируется как классическая задача регрессии — предсказания по множеству признаков некоторого скалярного значения.

Прямая модель: $(s_t, a_t) \rightarrow s_{t+1}$ (forward model). Задача предсказания следующего состояния по текущим состоянию и действию формализуется как определение параметров плотности распределения на множестве состояний. Данный тип модели встречается чаще всего и используется для прямого планирования (lookahead planning).

Обратная модель: $s_{t+1} \rightarrow (s_t, a_t)$. Задача предсказания того, какие были состояние и действие на предыдущем шаге, на основе текущего состояния. Такая модель используется для некоторых алгоритмов планирования в обратном направлении (от цели) [132].

Инверсная модель: $s_{t+1} \rightarrow (s_t, a_t)$. Задача предсказания действия, которое было совершено в некотором состоянии для перехода в следующее, также легко ложится в условия задачи регрессии. Такая модель используется в алгоритмах планирования по выборке, например, в RRT [367].

Рассмотрим классы моделей в зависимости от варианта аппроксимации, используемого в конкретном методе обучения с учителем.

Табличная параметрическая аппроксимация. Основным преимуществом в использовании множества параметров является независимость их количества от размера набора данных. Табличный вариант является наиболее простой реализацией, в которое для каждого возможного перехода в среде поддерживается своя запись. Например, для стохастических МППР может использоваться *табличная модель*

максимального правдоподобия [600]:

$$\hat{T}(s'|s,a) = \frac{n(s,a,s')}{\sum_{s'} n(s,a,s')},$$

в которой оценивается вероятность каждого перехода по количеству его наблюдения $n(s,a,s')$ в наборе данных. Естественно, что такие типы моделей не масштабируются на пространства состояний высокой размерности.

Функциональная параметрическая аппроксимация. Для высокоразмерных задач используется параметризация функции, определяющая некоторое общее функциональное семейство, в котором необходимо найти наилучшее приближение по имеющемуся набору данных. Наиболее популярными параметрическими семействами функций являются линейная регрессия, динамические байесовские сети, случайные леса и нейронные сети [132; 316; 441]. В последнее время наиболее популярны стали глубокие нейронные сети как наиболее общие функциональные аппроксиматоры. В связи со своей высокой вычислительной эффективностью нейросетевые аппроксиматоры отлично справляются с высокоразмерными пространствами и различными видами нелинейных зависимостей.

Непараметрическая аппроксимация. В непараметрических методах обучения с учителем напрямую сохраняются наблюдаемые данные для представления модели. Примером могут служить гауссовские процессы [220], удобные для оценки неопределенности модели. Однако их вычислительная сложность напрямую зависит от размера наблюдаемых данных, что в некоторых случаях существенно сужает область их применимости в высокоразмерных задачах.

Важным вопросом также является определение области пространства состояний, в которой гарантируется корректность модели, которую строит агент. Обычно в контексте этого вопроса различают два типа моделей.

- *Глобальные модели* аппроксируют динамику среды во всем пространстве состояний. Это наиболее часто используемый тип моделей для работы со всей собранной базой прецедентов, но, разумеется, в этом случае возникают основные сложности для достижения необходимого уровня обобщения.
- *Локальные модели* подразумевают точную аппроксимацию динамики только в некоторой локальной точке текущего момента времени и обновляются после завершения очередного этапа планирования с их использованием. Чаще всего такой подход применяется в задачах управления, где используется линейная аппроксимация динамики в окрестности текущего состояния [403]. Преимуществами таких моделей являются возможность использования более простых функциональных семейств, меньшее количество данных для аппроксимации динамики и лучшие показатели обобщения в локальной области по сравнению с глобальными моделями.

Обучение модели в условиях частичной наблюдаемости

Как и при обучении стратегии или функции полезности, одной из основных проблем при обучении модели среды является частичная наблюдаемость большинства сред, интересных с практической точки зрения. В таком случае для аппроксимации состояния s по наблюдениям o_i $\hat{s} = f(o_1, o_2, \dots)$, которое уже используется для предсказания динамики среды, применяются различные реализации функции f :

- *Агрегация наблюдений* по некоторому окну (framestack) предполагает конкатенацию n последних наблюдений агента для восстановления информационного состояния. При данном подходе основным вопросом является выбор размера такого окна и его адаптация в процессе обучения. Рассматриваемый табличный подход к задаче ограничен тем, что не используется генерализация по таким историям наблюдений и он может применяться только при небольшом размере окна n .
- *Доверительные состояния* были описаны в разделении планирования в условиях неопределенности 1.2.3. В этом случае модель разделяется на две части: модель наблюдений $P(o|s)$ и модель переходов в латентном пространстве $\hat{T}(s'|s, a)$. Формализация обучения в таком случае реализуется в рамках моделей пространства состояний, например скрытых марковских моделей (HMM, hidden Markov models), а для поиска параметров используется схема максимизации ожиданий (EM-алгоритм). Данный подход предполагает априорное знание о структуре пространства доверительных состояний, что так же не позволяет использовать его для многих высокоразмерных задач.
- Рекуррентные нейронные сети являются одним из наиболее популярных решений, как и в обучении с подкреплением без модели [326; 523]. В связи с тем, что параметры рекуррентной сети являются общими для всех временных шагов, размер модели не зависит от длины наблюдения, по которой аппроксимируются состояния. Данный подход также хорошо применим к высокоразмерным задачам за счет эффективных градиентных методов обновления параметров. Однако, с другой стороны, процесс обучения может приводить к взрыву или затуханию градиента, и это не позволяет моделировать долгосрочные причинно-следственные зависимости. Существует ряд подходов, как, например, обходные связи (skip connections) и модули агрегации наблюдений, которые до определенной степени позволяют преодолеть обозначенные проблемы [322].
- *Внешняя память* дает возможность сохранять наблюдения в отдельную базу прецедентов со специальными процедурами ее пополнения и извлечения информации. Такие методы как нейросетевая машина Тьюринга [296] предлагают дифференцирование таких процедур и их настройку в процессе обучения. Внешняя память в меньшей степени подвержена проблеме затухающих градиентов и успешнее моделирует долговременные зависимости.

С задачей аппроксимации информационного состояния тесно связаны способы представления состояний и моделирования динамики на множестве этих представлений. Более подробно это будет рассмотрено в разделе 1.3.2.

Многошаговые предсказания

Описанные выше модели динамики среды являются одношаговыми, то есть обучаются в задаче предсказания одного следующего шага. Для задачи планирования необходимо многошаговое предсказание действий и следующих состояний. С этой целью одношаговая модель применяется итеративно: на ее вход подается предсказанное ей же следующее состояние. Недостаток такого подхода заключается в том, что одношаговая модель, как правило, не оптимизируется на долгосрочные предсказания и получающаяся предсказанная траектория существенно расходится с реальной динамикой за счет накопления ошибки [603].

Для преодоления проблемы расхождения предсказаний и получения более качественных траекторий обычно используются две техники:

- модификация функции потерь: используется так называемая многошаговая (n -step) функция потерь [384; 386]; в этом случае модель все так же делает одношаговое предсказание, но функция потерь вычисляется на n ее последовательных предсказаниях, сравниваемых с n реальными наблюдениями из среды;
- специальные модели для n -шаговых предсказания – например, обучается модель в виде $\hat{T}(\hat{s}_{t+3}|s_t, a_t, a_{t+1}, a_{t+2})$ на текущем состоянии и цепочке из трех действий; в некоторых случаях предсказываются траектории целиком [436], но это требует в общем виде большего количества параметров для модели, хотя и приводит к стабилизации процесса обучения;
- отказ от фиксированного горизонта предсказания: в некоторых работах предлагается адаптивно выбирать количество шагов, на которое делает предсказания модель, на основе измерения степени уверенности предсказания для того или иного состояния [119].

1.3.2 Обучение представлений для модели среды

Абстракция состояний

В машинном обучении с подкреплением важную роль играет снижение размерности входных данных или, другими словами, эффективное представление наблюдений [594].

Качественное представление состояний и наблюдений позволяет делать корректные предсказания по модели и эффективнее обучать стратегию и функцию полезности.

Простейшим вариантом представления состояний является их агрегация [574], выделение главных компонентов (PCA) [466] или обучение простых базовых функций в МППР [416]. Но наиболее эффективным в настоящее время является использование глубоких нейросетевых кодировщиков, в том числе и для случая обучения модели динамики [115; 522].

При использовании нейросетевых аппроксиматоров модель декомпозируется на три составляющие:

- функция кодирования наблюдения $z_t = f^{enc}(o_t; \varphi)$ в латентное представление z_t с параметрами φ ;
- модель динамики в латентном пространстве $z_{t+1} = f^{trans}(z_t, a_t; \varphi)$, предсказывающая следующее латентное представление в зависимости от совершенного действия;
- функция декодирования латентного представления обратно в пространство наблюдений $s_{t+1} = f^{dec}(z_{t+1}; \varphi)$.

Данная схема соответствует классическому автокодировщику, используемому в различных генеративных задачах, с добавлением функции переходов.

В связи с тем, что латентное пространство, как правило, меньше, чем исходное пространство наблюдений или состояний, планирование в латентном пространстве более эффективно. Единственным ограничением здесь является то, что необходимо гарантировать, что предсказываемое латентное представление лежит в том же пространстве векторных представлений, что и кодируемые наблюдения. Существует два основных подхода для удовлетворения такого ограничения:

- глубокие модели пространства состояний, например, глубокие фильтры Калмана [365] и глубокие байесовские вариационные фильтры [218];
- добавление отдельного компонента к функции потерь, оценивающего близость предсказываемого латентного представления тому, которое получается кодированием истинного следующего наблюдения; наложение дополнительных ограничений на модель динамики в латентном пространстве в случае использования специальных процедур планирования. Например, при использовании линейно-квадратичного регулятора (LQR и iLQR) должна использоваться линейная модель динамики [245; 619].

Другим способом формировать эффективные представления является объектный подход, позволяющий выделять в наблюдении отдельные сущности (объекты) и идентифицировать типы взаимодействия между ними. Идея возможности использования объектного представления обосновывается наблюдениями за организацией высших когнитивных функций человека [586; 673]. В контексте машинного обучения с подкреплением данная идея была formalизована в виде объектных [230] и реляционных [275] МППР. Предполагается, что модели, использующие объектное представление, обладают более высокой степенью генерализации относительно новых условий в среде, так как есть возможность отделить информацию о самих объектах и их взаимодействий друг с другом от правил функционирования самой среды (например, законов физики в задачах управления).

Работа с объектными представлениями, как правило, обнаруживает две основные проблемы. Первая задача включает в себя собственно идентификацию и выделение объектов в наблюдении агента. Традиционно она решается путем применения либо специализированных моделей, обучающихся для обнаружения объектов [401; 553], либо моделей, обучающихся в самоконтролируемом режиме и в режиме без учителя [184; 442; 529]. Вторая проблема заключается в подборе способа моделирования взаимодействия объектов в латентном пространстве. Обычно для этого применяются графовые нейронные сети, в которых узлы отвечают за представление объектов (векторные представления или характерные признаки), а дуги — за функцию обновления этих представлений при взаимодействии этих объектов [353]. Существует ряд работ, демонстрирующих успешность применения таких моделей при построении моделей динамики среды: схематические сети [553], сети взаимодействий [330], нейросетевой симулятор физики [99], структурированные модели мира [353], COBRA [184] и др. В целом объектно-центрическое направление концентрируется вокруг набора способов структуризации нейросетевого латентного пространства для интерпретируемого сопоставления элементов этого пространства с объектами и фактами их взаимодействия [528].

Еще один способ формирования эффективных представлений — использование специализированных функций потерь. Известным подходом является переиспользование некоторых слоев нейросетевого кодировщика, применяемого для представления наблюдений в модели мира, в аппроксиматорах других функций, например, функции вознаграждения. Таким образом получается смешение целевых значений для ускоренного обучения представлений, получившее название *вспомогательных функций потерь* (auxiliary loss) [527].

В направлении модификации функции потерь эффективным подходом является *самоконтролируемая регуляризация*, то есть использование функций потерь, для которых явно не представлены целевые значения. Существует несколько вариантов их реализации.

- *Относительный эффект действий*: устранение незначимых элементов фона $s_{t+1} - s_t$ и предсказание изменяющихся признаков или движущихся объектов [264].
- *Учет непредвиденных обстоятельств* (contingency awareness): разделение факторов, определяющих поведение среды, на группу случайных, непредвиденных и на группу поддающихся контролю и предсказанию [195; 229]. Эффективное представление должно содержать информацию в первую очередь о факторах второй группы, например, путем использования модели обратной динамики $(s, s') \rightarrow a$ для предсказания действий, вызывающих определенный переход в среде [200].
- *Контрастивная функция потерь* опирается на схожесть или различие в группе наблюдений. Вводится некоторая дополнительная метрика близости состояний (например, динамическая — по количеству действий по переходу к данному состоянию из некоторого фиксированного [285]). Данная метрика вносит дополнительное ограничение на латентное пространство, делая его более удобным для задач предсказания и планирования [632].

Временная абстракция

Рассматриваемые ранее МППР являлись одноуровневыми в смысле используемых действий. Получающие траектории в таких МППР обычно подвержены проблеме редких и отложенных вознаграждений, когда необходимо связывать далеко отстоящие друг от друга события по совершению действия и получению вознаграждения за него (long-range credit assignment). Многие из таких траекторий имеют некоторые общие подпоследовательности, что приводит к идеи временной абстракции или, по-другому, использованию многоуровневого иерархического представления пространства действий [146; 315; 618]. Основной задачей в иерархическом обучении с подкреплением является определение высокоуровневого пространства действий, использование которого может существенно повысить эффективность выборок из среды [162] и снизить вычислительную эффективность процесса обучения [417].

Существует два основных подхода к описанию временных абстракций в пространстве действий. Первый известен как *метод умений* (options framework) [601]. Умением называется высокоуровневое действие $u = \langle I^u, \pi^u, \beta^u \rangle$, где $I^u \in S$ – множество состояний, с которых оно может начать выполнение, π^u – реализующая это умение стратегия и $\beta^u(s)$ – критерий останова, задающий вероятность завершения умения на множестве состояний.

Второй подход называется *целенаправленным обучением с подкреплением* (goal-conditioned) [205], или *универсальной аппроксимацией функции полезности* [629], в котором функция стратегии или полезности обуславливается на цель $g \in G$, задающей по сути некоторое абстрактное действие. Другими словами, целенаправленная функция полезности $Q^g(s, a, g)$ оценивает полезность действия a в состоянии s при попытке достижения цели g . Обучение модели при этом происходит с использованием параметризованной по цели функции вознаграждения (например, по евклидову расстоянию до целевого состояния [204]). Высокоуровневая стратегия в этом случае представляет собой последовательность подцелей.

Как при использовании метода умений, так и при целенаправленном обучении важнейшей задачей является эффективное выделение подпрограмм действий, или подцелей. Основные подходы к решению этой задачи следующие.

Графовая структуризация: обнаружение состояний, играющих роль *узких мест* (bottlenecks) в графе состояний среды. Обычно такими состояниями являются элементы МППР, соединяющие сильносвязные подграфы, что свидетельствует о том, что такие состояния являются важными с точки зрения исследования всего пространства состояний [428]. Среди известных методов по обнаружению узких мест – оценка частоты состояний в успешных траекториях [426], алгоритмы разрезания реконструированного графа МППР [572], оценка плотности исходящих и входящих ребер графа МППР [289]. Данный подход сложно масштабируем на высокоразмерные задачи.

Покрытия пространства состояний: отслеживание траекторий, достигших целевых состояний и позволяющих плотно покрыть пространство состояний. Обычно

такие подходы работают в предварительно кластеризованном пространстве состояний с некоторой аппроксимацией модели динамики [241; 415]. В целом такой подход работает и в латентном пространстве для высокоразмерных задач [286].

Информационный подход: предполагает использование определенных теоретико-информационных регуляризаторов при кластеризации пространства состояний и оценке распределения возможных подцелей [265; 381]. Среди таких регуляризаторов есть как классические методы оценки распределения шума [231; 634], так и формализация идеи предсказуемости траектории из определенных состояний (empowerment) [299].

Оценка по вознаграждению: предполагает, что эффективные подпрограммы действий должны приводить и к более высоким значениям вознаграждений, поэтому их определение можно встроить в непрерывный процесс оптимизации всей стратегии агента [429; 509]. Данные подходы предполагают дифференцируемость всей модели с интегрированными подцелями, например, критик-умений [143; 534] и феодальные сети [259]. Основной проблемой таких подходов является поддержание необходимого разнообразия множества умений без попадания в локальный минимум с единственным умением, решающим всю задачу целиком.

Априорные знания: использование дополнительной специфичной для задачи информации о возможных подпрограммах действий. Простейшим вариантом является заранее заданное множество подзадач [101] или использование некоторой эвристики (например, предположение, что все объекты должны быть подцелями [317]). Другой вариант предполагает выделение подцелей из множества демонстраций [116; 452].

1.3.3 Особенности планирования по обучаемой модели

Наличие истинной или обучаемой модели среды позволяет агенту использовать ее для ускорения процесса обучения собственной стратегии. Существует несколько вариантов использования модели, или, другими словами, интеграции планирования по модели и обучения стратегии. Выделяются четыре варианта такой интеграции.

Обучение с подкреплением с обучаемой моделью среды: одновременно обновляются и модель среды и функции стратегии/полезности. Простейшим примером является алгоритм Dyna [600].

Обучение с подкреплением с известной моделью среды: модель динамики точная и известна заранее, обучение стратегии/полезности происходит за счет планирования по модели. Типичным примером являются методы на основе MCTS, например, AlphaZero [423].

Планирование по обучаемой модели среды: обновляемая модель среды используется для локального планирования (на несколько шагов вперед) без обучения

глобальной функции стратегии или полезности. Типичный пример с планированием в латентном пространстве — Embed2Control [245].

Планирование без модели: процедура планирования и обучаемая модель объединены в общий оптимизационный цикл и неразделимы в плане оптимизируемых функций. Типичный пример — сети предсказания полезности (VPN, Value Prediction Networks) [475].

Следуя обзорной работе [443], выделим ключевые вопросы, связанные с процессом интеграции планирования по модели и обучения функций стратегии/полезности: в каком состоянии необходимо начинать процесс планирования, каким выбрать соотношение между собираемыми траекториями из среды и генерируемыми при планировании, какой алгоритм планирования выбрать и как объединить процессы обучения и планирования в цикл обучения.

Стартовое состояния для планирования

Обладая возможностью проводить выборку множества траекторий из модели, агент использует их для планирования своих действий. Для этого необходимо решить, в каком состоянии запускать процесс планирования, то есть, с каких состояний начинать выборку траектории из модели. Существует несколько вариантов выбора стартового состояния.

Случайный подход. Простейший вариант, когда выбираются случайные состояния из всего пространства состояний, чаще применяется в обычном динамическом программировании. При известной идеальной модели итеративные процедуры решения задачи динамического программирования (см. 1.2) дают хорошее приближенное решение. При большом пространстве состояний, впрочем, такой подход малоприменим, так как будет часто давать траектории, начальные состояния которых не достижимы из текущего.

Посещенные состояния. Для того чтобы работать с достижимыми траекториями, предлагается выбирать стартовые состояния из уже посещенных, как это реализуется в классическом подходе Dyna [600].

Приоритизированный подход. В некоторых случаях на множество достижимых состояний удается определить частичный порядок, по которому больший приоритет получают состояния, более подходящие для обновления стратегии в результате цикла планирования [447]. Данный подход похож на использование различных вариантов приоритизации в памяти прецедентов методов обучения с использованием отложенного опыта (см. DQN в разделе 1.1).

Текущее состояние. Наиболее частый подход — использование в качестве отправного текущего состояния в среде. В данном случае это эквивалентно локальному поиску лучшего решения в текущей области пространства состояний и позволяет использовать различные эвристики для сокращения пространства поиска при

планировании. Типичным примером такого подхода являются методы на базе MCTS [423].

Бюджет планирования

После определения отправной точки для планирования необходимо определить, сколько траекторий будут сгенерированы и какой они будут длины. Также важным вопросом является и то, как много шагов в реальной среде должно быть совершено, прежде чем модель достигнет определенного качества для нового цикла планирования. Так, в методе Dyna [600] используется до 100 шагов планирования после каждого шага в реальной среде. В методе PILCO [220] собираются данные с одного эпизода, а перепланирование затрагивает все решение, а не только локальную область пространства состояний. Экстремальным вариантом является пакетное [374] или *автономное обучение с подкреплением* [473], в котором опыт собирается однажды в некоторую память прецедентов, по которой уже необходимо выучить модель и стратегию агента без дополнительного взаимодействия со средой. Часто также используется вариант начального пакета опыта для предобучения модели среды, а затем обучение стратегии и дообучения модели идет в интерактивном режиме [245].

Собственно сами вычислительные затраты в процессе планирования складываются из двух составляющих: сколько раз запускается итерация планирования (или, иными словами, сколько раз выбирается новое начальное состояние для запуска выборки из модели) и сколько траекторий на определенный горизонт планирования выбирается из модели (собственно бюджет одной итерации планирования). Так, например, в алгоритме Dyna запускается 100 итераций с бюджетом (длиной траекторий, выбираемых из модели) в 1 шаг. В AlphaGo Zero наоборот, запускается только одна итерация MCTS, генерирующая 1600 траекторий по модели, каждая из которых с горизонтом планирования (максимальной глубиной поиска по дереву) 200 шагов. Итого сравнение бюджетом планирования Dyna и AlphaZero получается 1 к 3200. Отдельным случаем является вариант запуска процедуры планирования до полной сходимости к оптимальной стратегии на данных, полученных из модели [220].

Размер бюджета планирования определяет баланс между количеством шагов в реальной среде, которое выполняет агент, и количеством шагов в выбираемых из модели траекториях в процессе планирования. Крайними случаями в этом измерении являются безмодельное обучение с подкреплением с отсутствием шагов планирования и полный перебор с бесконечным бюджетом планирования. Очевидно, что эффективный баланс заключается в адаптивном поиске соотношения между быстрым реактивным поведением агента и медленным, но точным глубоком планированием. Одним из вариантов реализации такой адаптивной настройки является добавление небольшого штрафа за каждый шаг планирования в общей целевой функции [387]. Это позволяет получить гарантии того, что планирование должно вносить достаточный вклад в получаемое вознаграждение. Другой вариант, менее доменно зависимый, это обучение мета-контроллера, определяющего какой

бюджет выделить на текущем шаге [431]. В отличие от таких оптимизационных подходов, можно явно определять дисперсию оценки Q-функции и использовать эти данные для выбора соотношения реальных и планировочных шагов: чем больше дисперсия, тем больше должна быть выборка из модели [346]. С другой стороны, общепринято менее интенсивное использование модели на ранних шагах обучения в связи с ее низкой точностью.

Алгоритм планирования

Выбор алгоритма планирования, применяемого в обучении с подкреплением на основе модели, зависит от нескольких ключевых критериев, определяющих тип используемой модели и варианты проведения выборок из нее.

Дискретность и дифференцируемость модели. В большинстве классических подходов планирование применяется к дискретному множеству состояний, а сам процесс поиска происходит по организованному в дерево подмножеству МПР, получаемом из модели. В этом случае могут использоваться такие классические алгоритмы как вероятностный поиск [370], ограниченный в ширину и глубину поиск [188], поиск Монте-Карло [568] и Монте-Карло поиск по дереву [338; 423], минимаксный поиск [339] и одношаговый поиск [600]. Для более сложного пространства состояний в последнее время активное развитие получили варианты дифференцируемых моделей (например, в физических симуляторах [102; 247]), в которых функции переходов и вознаграждений могут быть продифференцированы по параметрам стратегии. В таком случае применимы дифференцируемые методы планирования: итеративное линейно-квадратичное планирование [619] в методе направленного поиска стратегии (GPS, Guided Policy Search) [404], градиенты полезности [383], модель гауссовых процессов в PILC, латентная нейросетевая модель динамики в Dreamer [237] и модели временных сегментов (Temporal Segment Models) [436]. Дифференцируемое планирование наиболее эффективно для непрерывных задач управления и робототехники, хотя и для дискретных задач в последнее время появились успешные примеры работы расширений метода Dreamer [421].

Направление планирования. Обычно процедура планирования применяется в режиме поиска в прямом направлении, то есть по ходу проигрывания эпизода. Однако в методах классического планирования популярным является обратное (backward) планирование от целевых состояний, когда используется обратная модель и по текущему состоянию предсказываются предыдущее состояние и совершенное действие $s' \rightarrow (s, a)$. В результате получается дерево поиска в обратном направлении и алгоритм планирования следует в направлении наибольших изменений оценок полезности [447]. Такой подход применялся в основном для небольшого пространства состояний и табличных подходов [208; 647]. При использовании аппроксимации есть

примеры работы с линейными функциями [240], моделью ближайших соседей [340] и нейросетевым подходом [199; 564; 583]. С нелинейными аппроксимациями обратный поиск ведет себя во многих случаях нестабильно.

Ширина поиска. В отличие от безмодельного обучения с подкреплением, когда каждый момент времени выбирается и выполняется одно действие, в процедуре планирования предварительный просмотр может проводиться на произвольную глубину и ширину. Иначе говоря, из модели могут выбираться произвольное количество траекторий произвольной длины. Простейшим вариантом является случай ширины 1, когда выбирается только одна траектория, к которой применяются обычное безмодельное обновления функции стратегии/полезности. Данный принцип применялся на базе метода глубокого градиента детерминированной стратегии (DDPG) [346], метода доверительных областей (TRPO) [391] и с обновлениями нормализованной функции преимущества (NAF) [198]. Типичным примером адаптивного подбора ширины поиска является MCTS подход, в котором для отсечения малоперспективных действий используется оценка верхней доверительной границы (UCB) [111]. Такая оценка неопределенности помогает в необходимых случаях вести поиск в глубину, вместо роста дерева в ширину. Наконец, крайним случаем является просмотр всех возможных вариантов действий в текущем состоянии, что происходит в классическом динамическом программировании.

Глубина поиска. Аналогично поиску в ширину, простейшим вариантом выбора длины траектории, генерируемой по модели, является единичная длина (как в Dyna). На противоположном полюсе находится методы PILCO и Deep PILCO [425], в которых из модели выбираются целиком эпизоды. Выбор среднего значения горизонта планирования также является результатом подбора некоторого баланса между неглубокими траекториями с низкой дисперсией и более глубокими выборками с низкой смещенностью оценки. Полезность не одношаговых траекторий была продемонстрирована в [320]. Однако важно отметить, что при неточной модели необходимо контролировать горизонт планирования, чтобы не получать слишком большие расхождения с реальной средой, и в некоторых случаях предсказание на один шаг будет являться более стабильным вариантом [646].

Оценка неопределенности. Так как во многих подходах модель является обучаемой, необходимо учитывать, что генерируемые по ней траектории могут отличаться от получаемых в реальной среде; данную неопределенность необходимо оценивать и учитывать при планировании. Простейший вариантом учета неопределенности модели — минимизация выборки из модели, что автоматически гарантирует нахождение вблизи реальных данных среды. Например, в Dyna используются одношаговые выборки с уже посещенных стартовых состояний, что даже при неточной модели не позволяет значительно отклониться от истинных траекторий. Более общий подход — явное введение ограничение на сохранение близости плана текущей стратегии агента, как это происходит в методе GPS. Однако чаще всего используется техника распространения по пространству состояний

измеренной некоторым способом неопределенности, возникающей из-за отклонения от наблюдаемых данных. Аналитический вариант измерения неопределенности возможен при использовании таких простых моделей, как гауссовые процессы в PILCO. Для нейросетевых моделей используется метод на основе выборок. За счет выборки в Deep PILCO [425] и в PETs [216] оцениваются параметры и моменты распределений, используемых в этих моделях наряду с нейросетевыми функциями динамики. Оценка неопределенности также может использоваться для регулирования горизонта планирования. Так, в методе STEVE [545] предлагается перевзвешивать целевые значения полезности на разных шагах планирования в соответствии с неопределенностью их оценки по модели динамики, что может стабилизировать обучение.

1.3.4 Интеграция планирования и обучения в едином цикле

На рисунке 1.11 изображена общая схема различных вариантов интеграции процессов планирования и обучения в единый цикл обучения агента. До этого основное внимание было уделено использованию модели в планировании (стрелка a). Сейчас же будут рассмотрены остальные связи модуля планирования с другими элементами цикла обучения: направление поиска при планировании за счет информации из функции стратегии и полезности (стрелка b), использование результатов планирования для обновления функций стратегии/полезности (стрелка c), использование результатов планирования для выбора действия в реальной среде (стрелка d).

Направление планирования

Функция стратегии и полезности содержит большое количество информации о свойствах среды, которую можно использовать для сокращения пространства поиска при планировании. Наиболее удобной является априорная информация, содержащаяся в функции полезности (value priors). Она может быть интегрирована в процесс планирования за счет обновления по траектории(бутстрепинга) значения полезности для предотвращения излишнего поиска в глубину. Такая техника используется во многих алгоритмах обучения на основе модели [338; 339; 423].

В итерации планирования может использовать и обучаемая функция стратегии. Так, в AlphaGo Zero вероятностное распределение на множестве действий, определяемое по функции стратегии, используется как априорный множитель в компоненте UCB в MCTS-планировании. Это дает больше бонусов к исследованию среды за счет более

вероятных по текущей стратегии агента действий. В GPS новые траектории, получаемые по модели, штрафуются при значительном отклонении от траекторий, генерируемых по текущей стратегии. В работе [214] текущая стратегия агента используется для определения начального состояния для осуществления новой итерации планирования, что является одной из форм приоритизации стартовых состояний.

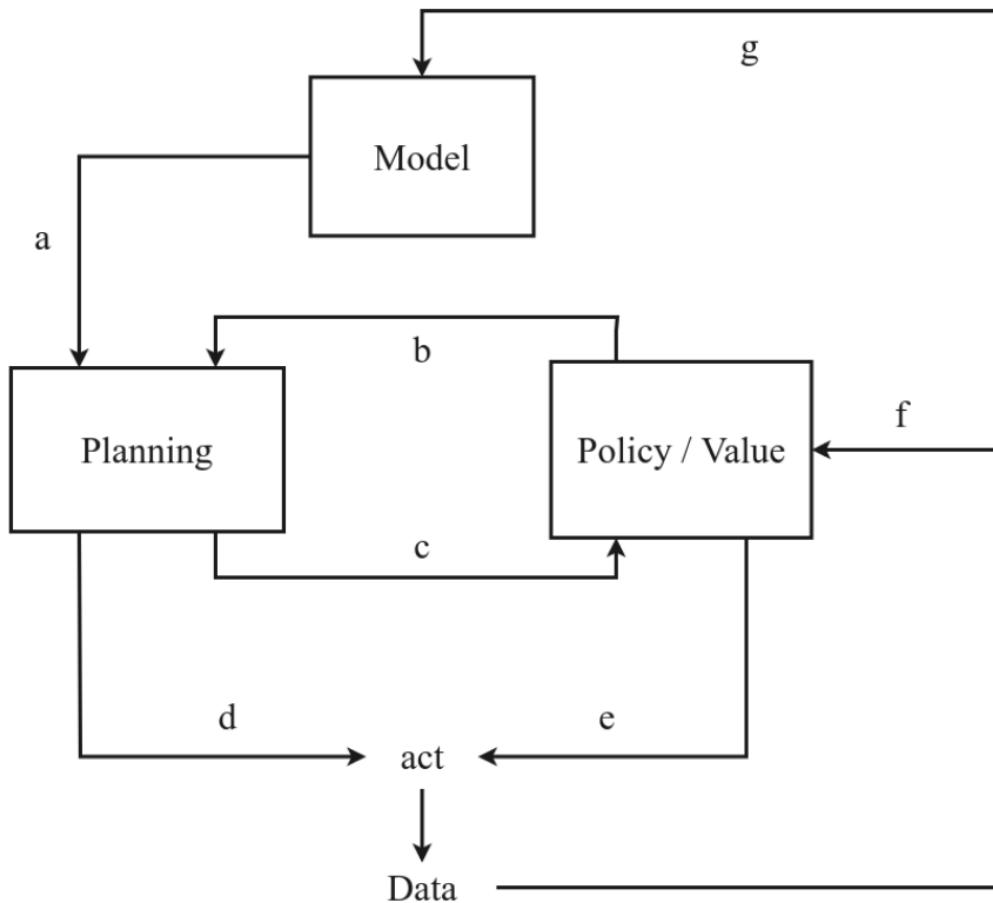


Рисунок 1.11 — Общая схема интеграции обучения и планирования (адаптировано из [443]).

Использование результатов планирования

Методы обучения с подкреплением на основе модели реализуют глобальную аппроксимацию оптимального значения функций полезности или стратегии. Результат планирования в таких алгоритмах может быть использован для обновления этого глобального приближения. Возможны две реализации такого обновления: обновление за счет поиска при планировании нового значения целевой переменной для обучения или регуляризация функции потерь.

Как и в случае обратного использования результатов обучения, в соответствии с архитектурой «актора-критика» возможно использование результатов планирования при обновлении функции полезности или функции стратегии. В первом случае типичным

выбором для целевого значения является оценка полезности состояния или действия в корне дерева поиска. Эта оценка зависит от стратегии обратного распространения информации по дереву и может быть реализована как по актуальному, так и по отложенному опыту. Для методов, которые выбирают из модели только один шаг (то есть используют единичную ширину поиска), таких как Dyna, используется классическое целевое значение Q-обучения (одношаговое, с отложенным опытом). Для процедур планирования, в которых учитывается множество действий в состоянии (которые реализуют поиск в ширину и, возможно, вглубь), существует возможность комбинировать различными способами распространение информации по дереву с актуальными и отложенными опытами. Так, в [648] проведен обзор различных вариантов обратного распространения в древовидном поиске. После построения целевого значения аппроксиматор полезности обычно обучается на основе среднеквадратичной ошибки (MSE) [158]. Однако возможны и другие варианты, например, функция потерь в виде кросс-энтропии между мягким максимум Q-значений, полученных в результате планирования, и Q-значений, получаемых из нейросетевой аппроксимации [190].

При обновлении функции стратегии также возможны различные варианты реализации целевого знания и функции потерь в зависимости от типа используемой процедуры планирования. Например, в AlphaGo Zero используется планирование MCTS, и целевое значение создается путем нормализации количества посещений в корневом узле. Затем нейросетевой аппроксиматор стратегии обучается на основе кросс-энтропийной функции потерь, определенной на таком распределении. В [214] применяется та же идея с однократным кодированием наилучшего действия, в то время как в [113] описывается реализация функции потерь, работающая как с дискретными, так и с непрерывными выходами аппроксиматора стратегии. В качестве другого подхода в GPS минимизируется дивергенция Кульбака-Лейблера (KL) между запланированной траекторией и результатами, полученными в аппроксиматоре стратегии. Некоторые подходы к дифференцируемому планированию также непосредственно обновляют дифференцируемое глобальное представление [220]. Кроме того, стратегия может обучаться на основе оценки функции полезности. Например, в методе поиска градиента стратегии (PGS) [503] используется теорема о градиенте стратегии [650] для ее обновления на основе оценок значений полезности в дереве.

Большинство из вышеперечисленных методов строят целевые значения полезности или стратегии в корне поиска. Однако, конечно, возможно формирование целевых значений на более глубоких уровнях дерева [158]. Это позволяет извлечь больше информации из цикла планирования. Во многих работах полезность или стратегия обновляются на основе как выбранных из модели, так и на реальных данных. С другой стороны, в ряде исследований функции полезности и стратегии обучаются исключительно на основе данных, полученных из модели [303; 391], используя реальные данные только для обучения модели динамики.

Необходимо обратить внимание, что стрелки b и c на рисунке 1.11 образуют замкнутый подцикл в общем цикле интеграции. В последнее время наблюдается большой интерес к особенностям работы этого подцикла, который перемежает планирование, основанное на априорной информации, полученной по функции стратегии или полезности (стрелка b), и

обучение этих функций, основанное на результатах планирования (стрелка с). Успешным алгоритмом в этом классе является AlphaGo Zero, который служит примером многошагового приближенного динамического программирования в реальном времени (MSA-RTDP) [243]. MSA-RTDP расширяет классические идеи DP, используя многошаговое прогнозирование по модели, обновляя аппроксиматоры полезности или стратегии и оперируя траекториями, полученными из среды (в режиме реального времени). В [243] теоретически исследуется концепция MSA-RTDP и показывается, что более высокая глубина планирования снижает сложность выборки в реальной среде за счет увеличения вычислительной сложности. Несмотря на то что это интуитивный результат, он доказывает, что планирование может привести к принятию более обоснованных решений в реальном мире за счет увеличения времени на прогноз (основанный на модели). Кроме того, чередующееся планирование и обучение также могут привести к более стабильному обучению.

Выбор действия после планирования

Результаты планирования могут также использоваться для прямого выбора действий в реальной среде. В то время как в безмодельном обучении с подкреплением всегда используется аппроксиматор функций полезности или стратегии для выбора нового действия в среде (рис. 1.11 , стрелка е), модель среды позволяет выбирать действия непосредственно из выходных данных процедуры планирования (рис. 1.11 , стрелка д). Некоторые методы используют планирование только для выбора действий, а не для обновления функций полезности или стратегии [568; 608], например, потому, что обновления целевых значений в результате планирования могут содержать существенную неопределенность. Впрочем, многие методы фактически сочетают оба вида использования выборок из модели [113; 133; 423].

Выбор действий в реальной среде может происходить различными способами. Простейшим вариантом является «жадный» способ выбирать лучшее действие из плана. Этот подход типичен для методов, которые планируют на основе обучаемой модели. Характерным примером в этой группе являются управление с предсказанием по модели (МРС) или методы управления с расширяемым горизонтом. В МРС «жадное» действие выбирается в результате k-шагового предварительного поиска, которое потом применяется в среде. Наблюдаемое истинное следующее состояние используется в новой итерации той же процедуры. Фактический алгоритм планирования в МРС может варьироваться. Примерами могут служить iLQR [245], прямое оптимальное управление [216; 461], алгоритм Дейкстры [391] или повторное применение обратной модели [398]. МРС устойчив к изменяющимся ограничениям на состояние и пространство действий [347] и особенно популярен в робототехнике и задачах управления.

Необходимо отметить, что не обязательно выполнять «жадное» действие после завершения цикла планирования. Как вариант, практикуется введение дополнительного

шума для улучшения исследования среды (например, шум Дирихиле в [423]). Также есть возможность явно включить критерии, улучшающие исследование среды в процесс планирования, которые реализуют принцип *планирования исследования среды* [497; 498]. Например, в [209] исследуется применимость полезности совершенной информации (VPI), оцениваемой на основе модели, к выбору действия, имеющего наибольший потенциал для изменения «жадной» стратегии. В самом деле, планирование может определить последовательности действий, которые выполняют глубокое исследование в направлении новых областей пространства состояний в высокими вознаграждениями, которые одношаговые методы исследования среды не смогли бы идентифицировать вследствие неустойчивости поведения [211].

1.4 Выводы

Современные методы обучения с подкреплением и автоматического планирования в расширенной постановке являются разными подходами к одной проблеме последовательного принятия решений в условиях неопределенности. В обоих подходах имеется техника накопления опыта в условиях неопределенности. Однако, с точки зрения математической постановки, задача обучения с подкреплением сводится к задаче оптимизации и поиска оптимального множества параметров в функциональном пространстве, а задача планирования — к задаче поиска в дискретном пространстве возможных обобщенных последовательностей действий (на графе или в дереве). Мощным инструментом последовательного принятия решений в условиях неопределенности является гибридной подход — обучение с подкреплением на основе модели или планирование с обучаемой стратегией. Существует несколько вариантов интеграции планирования и обучения, однако все они обладают определенными недостатками, в то время как системной архитектуры интеллектуального агента, в которой была бы возможность реализовать комплексный универсальный вариант интеграции, в настоящее время не существует.

Отдельно необходимо отметить различные подходы к представлению состояний в обучении и при планировании. Если в первом случае существуют эффективные аппроксиматоры, работающие с высокоразмерными пространствами состояний (например, с изображениями), то во втором обычно идет работа с исходным признаковым описанием состояния или наблюдения. Для создания эффективных методов обучения с подкреплением на основе модели необходимы гибридные или так называемые нейросимвольные подходы в кодировании представлений как для использования в модели среды, так и для задания наблюдения в обучении и при планировании.

В дальнейших главах настоящей исследовательской работы будет рассмотрена возможность построения нейросимвольной архитектуры генерации поведения когнитивного агента, в которой будут сочетаться методы планирования по модели мира и методы обучения стратегии, в основном, в реализации «актор-критик». Будет предложена конкретная

реализация данной архитектуры под названием NSLP с использованием различных механизмов объектно-центричного представления состояний и динамики среды.

Глава 2. Нейросимвольная архитектура для планирования и обучения

В данной главе представлены две архитектуры управления воплощенным когнитивным агентом общего назначения: STRL (иерархическая архитектура со стратегическим, тактическим и реактивным уровнями генерации поведения), в которой раскрываются взаимосвязи между компонентами, реализующими перцептивный анализ ситуации, управление движением и стратегическое планирование, и NSLP (нейросимвольная архитектура для планирования и обучения), в которой уточняется возможность реализации одновременного обучения и планирования на стратегическом уровне архитектуры STRL. Основное внимание уделено уточнению постановки задачи принятия решений в условиях неопределенности, которая в общем виде была представлена в главе 1.3, а также формальным свойствам реализации процесса обучения в архитектуре NSLP. В заключение главы изучается возможность концептуального планирования и генерации подцелей с помощью предобученных языковых моделей на концептуальном уровне архитектуры NSLP. Основные результаты, представленные в данной главе, были опубликованы в работах [91] (раздел 2.1), [13; 64; 88; 89; 95; 98] (раздел 2.2), [61; 96] (раздел 2.3), [62; 66] (раздел 2.4).

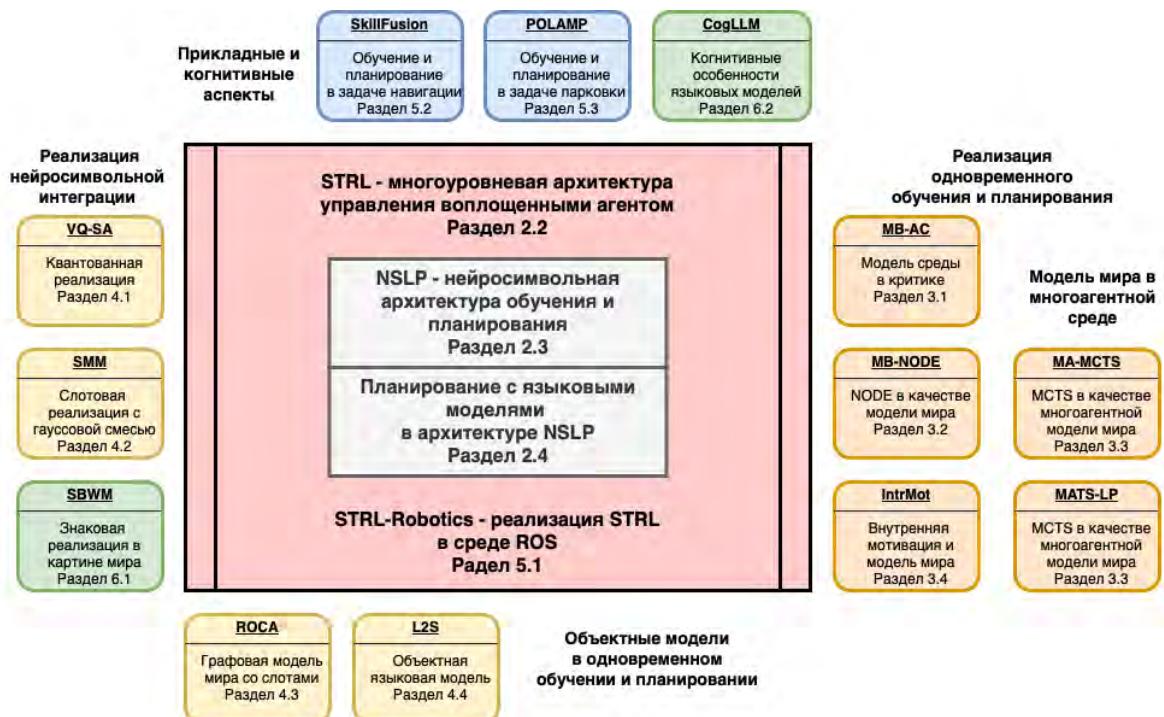


Рисунок 2.1 — Взаимосвязь архитектур STRL и NSLP с программно-алгоритмическим инструментарием, представленным в дальнейших главах диссертационного исследования.

На рисунке 2.1 представлена взаимосвязь описываемых в данной главе архитектур STRL и NSLP с описываемыми далее в данном диссертационном исследовании программно-алгоритмическим инструментарием. Один из вариантов полной реализации с помощью программного обеспечения ROS представлен в разделе 5.1 (STRL-Robotics).

Реализация нейросимвольной компоненты архитектуры NSLP подробно представлены методами VQ-SA, SMM и SBWM. Реализация концепции одновременного обучения и планирования в архитектуре NSLP изложена с помощью методов MB-AC, MB-NODE, MA-MCTS и MATS-LP (глава 3). Использование объектного принципа нейросимвольной интеграции в обучении и планировании иллюстрируется методами ROCA (раздел 4.3) и L2S (раздел 4.4). Прикладные и когнитивные аспекты освещаются в главах 5 и 6.

2.1 Когнитивные архитектуры

Создание когнитивных архитектур — один из наиболее ярких примеров системного подхода в искусственном интеллекте при решении задачи генерации поведения интеллектуальным агентом. Основной идеей этого направления является максимально точное моделирование на системном уровне психологических и нейрофизиологических характеристик человека при реализации им основных когнитивных функций, таких как приобретение и использование знаний, и генерация поведения. Таким образом, можно сказать, что подход когнитивных архитектур заключается в моделировании когнитивных функций человека с той или иной степенью психологической или биологической правдоподобности. Системность этого подхода проявляется в выделении ключевых подсистем, участвующих в работе моделируемой функции по данным психологов или нейрофизиологов, построении ее математической модели и затем рассмотрение всех видов взаимодействия этих подсистем. Ключевая гипотеза заключается в том, что именно в результате взаимодействия подсистем и возникает системный эффект — проявление этой функции (проведение рассуждения или формирование последовательности принятия решений).

Отсюда вытекают основные задачи или этапы при создании когнитивной архитектуры:

- определение списка моделируемых функций (распознавание, категоризация, внимание, планирование поведения, обучение, рефлексия, рассуждения и т.д.);
- определение набора моделей или теорий «первого уровня» (например, нейрофизиологические данные о строении неокортекса или когнитивная теория внимания), которые будут служить источником основных гипотез при моделировании выбранных функций;
- на основании сформулированных гипотез выделение основных подсистем, которые определяют работу выбранных функций;
- построение математических моделей работы выделенных подсистем и разработка «протокола коммуникации» подсистем в рамках общей архитектуры;
- добавление недостающих компонент для погружения в экспериментальную среду, в которой будет проводиться тестирование созданной архитектуры, т.е. сравнение работы моделируемых ей когнитивных функций с тем, как эти функции работают в человеческом мозге.

Понятие когнитивной архитектуры синонимично понятию интеллектуального агента и подразумевает воплощение полного агента, способного самостоятельно действовать в некотором окружении, достигая определенные цели или удовлетворяя свои потребности. Когнитивная архитектура включает набор систем памяти, которыми обладает человек: рабочая, процедурная, семантическая, эпизодическая и т.д. В основе ее работы лежит когнитивный цикл, включающий как минимум восприятие, осмысление, принятие решений и контроль над выполнением действий, причем цикл может выполняться параллельно и асинхронно. Можно сказать, что представленная на рисунке ниже модель содержит минимальный набор блоков, которыми должна обладать некоторая архитектура, чтобы называться когнитивной. Данный набор блоков нельзя назвать однозначным. К примеру, объединение блоков памяти с блоками обучения, хотя и может быть мотивировано нейрофизиологией и когнитивной психологией, уменьшает детальность модели (см. рисунок 2.2).

В том случае, когда когнитивная архитектура включает достаточно широкий спектр моделируемых когнитивных функций и предназначена для решения широкого круга принципиально различных задач, можно считать, что данная архитектура является реализацией системы универсального искусственного интеллекта (AGI). Однако многие когнитивные архитектуры не претендуют на высокую степень универсальности и предназначены для моделирования вполне конкретной функции, например, приобретения знаний без учителя (HTM [280]), восприятия и внимания (ART [301]), формирования правил поведения на основе изображений (Clarion [598]), памяти (ACT-R [114]), эмоций (eBICA [547]) и т.д.

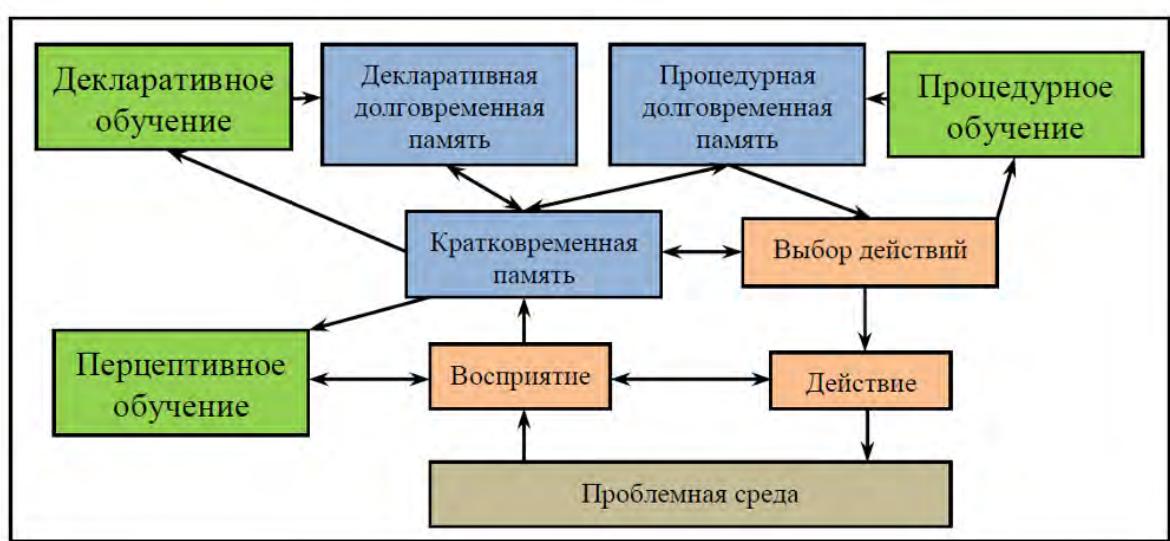


Рисунок 2.2 — Основные функциональные блоки, характерные для многих когнитивных архитектур

Как и многие другие методы, опирающиеся на системный подход, когнитивные архитектуры не всегда имеют программную реализацию и часто остаются на концептуальном уровне, на котором описаны основные подсистемы, их функции, без конкретной формализации. До программной реализации и, тем более, до демонстрации работы в

сложных условиях доведены не все архитектуры. Многие из них остаются на символическом уровне, не решая ключевую проблему универсального искусственного интеллекта — проблему привязки символов. Пик интереса к когнитивным архитектурам пришелся на 2000–2010 гг., когда наблюдалась максимальная активность в разработке большого количества различных вариантов. В репозитории BICA¹ собрана довольно обширная подборка существовавших на этот момент наиболее проработанных архитектур. Среди них можно выделить наиболее старые и универсальные: Soar (Лэйрд, Розенблум и Ньюэлл [372]) и ACT-R (Андерсон и Леберь [131]), — которые развиваются до настоящего времени и которые, в целом, можно назвать в некотором приближении системами AGI. Данные примеры являются психологически правдоподобными. К примерам биологически правдоподобных архитектур относятся ART (Гроссберг [169]) и HTM (Георг и Хокинс [280]), которые, естественно, претендуют на моделирование менее широкого спектра функций, но, тем не менее, более формализованных и системных, чем широкомасштабные модели мозга, типа Spaun (Элиасмит [595]).

На текущий момент необходимо отметить снижение интереса к когнитивным архитектурам, что вызвано двумя основными причинами. Первая заключается в слабой интеграции субсимвольных и символьных подходов, особенно в настоящее время, когда успехи субсимвольного нейросетевого подхода приводят к мысли, что «символьная добавка» не играет ключевой роли для моделирования даже сложных функций (например, рассуждения на основе глубоких нейронных сетей). Вторая проблема состоит в том, что существует определенная произвольность в выделении ключевых подсистем, зачастую из-за большой неточности теорий «первого уровня». Широкое распространение получает подход самоорганизации, в котором предполагаемая архитектура подсистем не задается заранее, а является результатом развития, обучения, всей системы.

Символьный подход в искусственном интеллекте базируется на гипотезе Саймона и Ньюэлла: «Физическая символьная система имеет необходимые и достаточные средства для произведения основных интеллектуальных операций». Символьный подход продемонстрировал свою мощь в таких задачах как планирование и рассуждения. Однако их ограниченность на данный момент, особенно при работе с сенсорными неструктуризованными данными, очевидна. С другой стороны, очевидна и необходимость создания гибридного подхода, интегрирующего символьные методы (в особенности, системы логического вывода и системы, основанные на правилах), наряду с современными методами машинного обучения, в том числе и нейросетевыми.

Отмеченные выше недостатки в создании классических когнитивных архитектур приводят к построению гибридных архитектур, которые зачастую не имеют выделенной структуры подсистем. В качестве примера приведем архитектуру STRL.

Основой моделируемой функцией в этой архитектуре является управление сложным физическим объектом (например, мобильной платформой с манипулятором или беспилотным летательным аппаратом) в условиях коллективного взаимодействия или взаимодействия с оператором при решении общих задач по перемещению и манипулированию с объектами. Она

¹<https://bica.ai/architectures/>

состоит из трех уровней: стратегического, тактического и реактивного (архитектура STRL: strategic, tactical, reactive layered). Стратегический уровень ответственен за реализацию высокоуровневых интеллектуальных функций. На нем используется знаковая реализация механизма нейросимвольной интеграции представления знаний и осуществляется обмен сообщениями с остальными участниками коалиции. Тактический и реактивный уровни содержат модули, поддерживающие эти процедуры и транслирующие их управление в низкоуровневые управляющие сигналы. Подробнее см. раздел 2.2.

На стратегическом уровне для реализации функции обучения концептуальным планам, планирования сложного поведения, распределения ролей, целеполагания и рефлексии было предложено использовать теорию знаковой картины мира [44]. Данный подход основан на культурно-историческом подходе Выготского и теории деятельности Леонтьева. Основным активным элементом картины мира предлагается использовать понятие знака. Знак, как это принято в деятельностном подходе, обладает четырьмя компонентами: образом (чувственной тканью), значением, личностным смыслом и именем. Модель картины мира основана на взаимодействии четырех типов семантических сетей, вершинами в которых являются компоненты знаков. Когнитивные процессы в модели реализуются благодаря распространению активности, механизмы которой различны для каждого типа семантической сети на компонентах знака. Элементарный этап работы каждой когнитивной функции заключается в остановке распространения активности и связывании активных узлов каждой из сетей в знак, который в зависимости от задачи является знаком-целью, знаком-ситуацией или знаком-ролью. Таким образом, в модели знаковой картины мира когнитивный процесс представляет собой последовательность активации (или образования) знаков. Существенным отличием предлагаемого механизма распространения активности от существующих в работах по искусственному интеллекту является взаимодействие четырех типов сетей, в отличие от «односетевых» моделей нейронных сетей, семантических сетей и сетей Петри. В семиотической сети реализуются функции приобретения знания: формирования компонент знака на основе методов машинного обучения с подкреплением и без учителя, а затем связывания этих компонент в единую структуру и последующее именование.

Когнитивные науки являются источником в некоторых случаях хорошо обоснованных подробных неформальных моделей работы ряда когнитивных функций, которые обязательно должны быть реализованы в полноценных системах универсального искусственного интеллекта.

2.2 Архитектуры STRL и STRL2

Одной из наиболее перспективных и активно-развивающихся междисциплинарных областей, предполагающих использование систем искусственного интеллекта общего назначения, является область, связанная с проектированием и разработкой беспилотных

транспортных средств различного типа и назначения, в частности – малых беспилотных летательных аппаратов (БПЛА). Среди последних широкое распространение получили аппараты вертикального взлета и посадки, выполненные по мультироторной схеме: AscTec Hummingbird², Parrot AR.Drone [610]³, mikroopter⁴, 3DR IRIS⁵ и др. Упомянутые БПЛА представляют собой унифицированные платформы, которые оборудованы необходимыми датчиками, органами управления, бортовыми вычислителями (полетными контроллерами), а также встроенным программным обеспечением, которое используется для автоматизации отдельных режимов полета и маневров (взлет, посадка, полет в плоскости, изменение угла рыканья и т.д.).

Базовое бортовое программное обеспечение обычно допускает возможность интеграции сторонних модулей по открытym протоколам и программным интерфейсам, и эта возможность активно используется исследователями. Последние разрабатывают и программно реализуют различные методы управления и навигации (см. например обзор в [349]) для их последующего включения в единую систему управления БПЛА [693]. Весьма актуальной при этом является задача организации взаимодействия модулей (разработки архитектуры собственно системы управления общего назначения).

Системы управления, представляющие наибольший интерес в настоящее время, являются интеллектуальными системами управления [667] (ИСУ). ИСУ – это такая система, которая в автоматическом режиме способна к решению нетривиальных, интеллектуальных задач, включая планирование, целеполагание, прогнозирование, формирование коалиций, и тем самым способна обеспечить высокий уровень автономности объекта управления (БПЛА). Ряд вышеперечисленных задач не решается в рамках известных подходов искусственного интеллекта и требует привлечения новых методов, развиваемых в рамках компьютерного когнитивного моделирования, где используются модели когнитивных функций человека. Исследователями в этой области разрабатываются отдельные элементы интеллектуальных систем управления, которые, однако, не являются самодостаточными. Для построения полноценной системы управления реальным сложным техническим объектом необходима интеграция методов решения указанных выше интеллектуальных задач с методами решения таких задач как локализация, планирование траектории, следование по траектории, т.е. задач более низкого уровня. Такая интеграция и является предметом исследования в данном разделе.

В разделе рассматриваются такие средне- и низкоуровневые задачи как стабилизация БПЛА, планирование траектории, а также предлагаются методы решения высокоуровневых задач (построение картины мира, планирование поведения и т.д.), опирающиеся на модели соответствующих когнитивных функций, подтвержденные психологическими и нейрофизиологическими данными. В результате рассмотрения предлагается многоуровневая архитектура – *STRL архитектура* (от англ. Strategic, Tactical, Reactive, Layered)

²<http://www.asctec.de/en/uav-uas-drone-products/asctec-hummingbird/>

³<http://ardrone2.parrot.com/>

⁴<http://mikroopter.de>

⁵<http://3drobotics.com/iris/>

интеллектуальной системы управления сложным техническим объектом (например, БПЛА), использование которой позволит автоматизировать управление не только отдельным БПЛА, но и их коалициями при решении широкого круга сложных, комплексных задач в различных средах.

Основные результаты, изложенные в данном разделе, представлены в публикациях [13; 64; 88; 89; 95; 98].

2.2.1 Архитектура STRL для управления группой сложных технических объектов

Известно множество различных подходов к созданию систем управления сложными техническими объектами, в том числе БПЛА, и общая классификация архитектур таких систем затруднена [546]. Одним из возможных способов классификации является использование следующих критериев: уровень иерархии – одноуровневые, двухуровневые, многоуровневые и т.д., и тип функциональной декомпозиции – явная и неявная. При указанном способе классификации в отдельный класс архитектур ИСУ БПЛА обычно выделяют плоские, одноуровневые архитектуры с явной функциональной декомпозицией. В этом случае система управления представляет собой набор модулей не связанных никакой иерархией, и каждый модуль решает вполне определенный набор задач. Интеллектуальные функции при этом распределены по всей системе, т.е. каждый модуль может реализовывать часть из них. Пример подобной архитектуры описан, например, в [187]. К другому классу архитектур относятся многоуровневые архитектуры с неявной функциональной декомпозицией. Каждый уровень архитектуры состоит из элементов, абстрагирующих отдельные сущности управления (подсистема БПЛА, сам БПЛА, группа БПЛА, группа групп БПЛА и т.д.), и каждый элемент состоит из фиксированного числа идентичных модулей с неявной спецификацией. Наиболее показательный пример подобного рода архитектур – 4D/RCS [122]. В рамках 4D/RCS выделяются следующие неявно специфицированные модули (или, в оригинальной терминологии, – функциональные процессы), которые реализуют каждую функцию системы: планирование, моделирование, обработка сигналов, оценивание свойств. Неявность спецификации заключается в следующем (рассмотрим на примере модуля "планирование"). На более высоких уровнях иерархии 4D/RCS под "планированием" понимается планирование поведения (т.е. поиск планов в пространстве формализованных ситуаций, возможностей, целей объекта управления), на более низких – планирование траектории, ещё на более низких – управление (т.е. поиск планов в пространстве управляющих сигналов БПЛА). При таком подходе интеллектуальные функции сконцентрированы на верхних уровнях иерархии и определены неявно.

Наконец, самыми распространенными являются многоуровневые архитектуры с явной функциональной декомпозицией. В этом случае функциональность каждого модуля явно определена, и модули разделены на уровни: чем выше уровень, тем больше возможностей абстрагирования от значений сигналов сенсоров БПЛА используется в соответствующем данному уровню представлении знаний. Задачи, решаемые на более высоких уровнях системы, таким образом, считаются более сложными, интеллектуальными. Обычно в области робототехники и беспилотных транспортных средств выделяют три уровня управления [98]. В качестве примера соответствующих архитектур можно назвать ATLANTIS [272], 3T [273], Aura [134] и другие. Упомянутые архитектуры подразумевают реактивную выработку управляющих сигналов для органов управления (реактивное управление) на нижнем уровне, и решение всех остальных (делиберативных) задач на верхнем, в то время как средний уровень обеспечивает межуровневые взаимодействие.

Описываемая в данном разделе архитектура, STRL, также является трехуровневой, однако ее средний уровень выполняет не только роль интерфейса между верхним и нижним уровнями, но и инкапсулирует значительную часть функционала, связанного с пространственной навигацией. К значимым отличиям данной архитектуры относятся также: поддержка механизмов коллективного планирования и распределения ролей, поддержка механизмов трансляции ограничений на динамику движения в геометрические ограничения на форму траектории, планирование с учетом этих ограничений и ряд других. Далее опишем STRL архитектуру более подробно.

Уровни управления архитектуры STRL

В настоящем разделе описывается предлагаемая диссертации архитектура системы управления сложными техническим объектом (например, мобильная платформа или БПЛА), состоящая из трех уровней: стратегического, тактического и реактивного (архитектура STRL: strategic, tactical, reactive layered). Система, построенная по предложенной архитектуре, управляет поведением сложного технического объекта (например, БПЛА) в условиях коллективного взаимодействия при решении общих и частных задач. Стратегический уровень является ответственным за реализацию высокоуровневых интеллектуальных функций. На нем используется знаковый механизм нейросимвольной интеграции для построения гибридного представления знаний [677] и осуществляется обмен сообщениями с остальными участниками коалиции и с оператором задачи. Тактический и реактивный уровни содержат модули, поддерживающие эти процедуры и транслирующие их управление в низкоуровневые управляющие сигналы. Общий вид архитектуры представлен на рисунке 2.3, где показаны основные модули, выделенные в соответствии с функциональными возможностями каждого из уровней, а также взаимодействие между ними (вид и направление передаваемых данных).

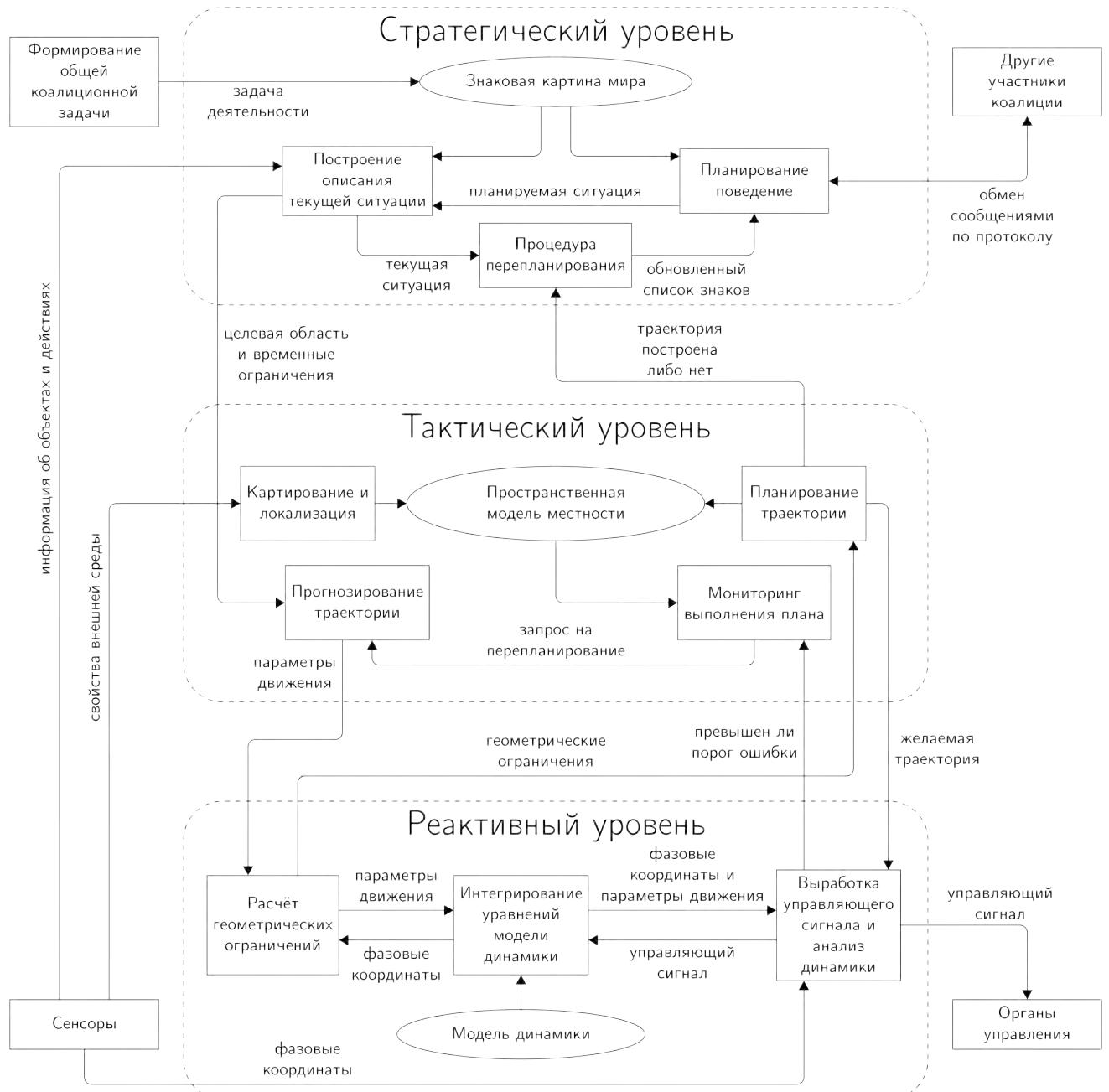


Рисунок 2.3 — Архитектура STRL системы управления группой сложных технических объектов.

Основной задачей управления на стратегическом уровне является построение согласованного плана поведения участника совместной деятельности. Система управления участника коалиции строит и поддерживает в актуальном состоянии картину мира, которая является уникальной для каждой системы благодаря как характерным свойствам, ограничивающим множество доступных системе управления действий, так и благодаря различию в строении компонент элементов картины мира. Общими компонентами всех участников коалиции являются компоненты значения элементов картины мира.

Построение плана поведения происходит с помощью обмена сообщениями с другими участниками коалиции. На каждом этапе выполнения плана происходит обновление описания текущей ситуации, в том числе, и с участием информации, поступающей от сенсоров. Из описания текущей ситуации планирования выделяется пространственно-временная информация, которая формирует задание, поступающее на тактический уровень управления и содержащее пространственное описание целевой области, в которую необходимо переместить объект управления, а также временные ограничения на её достижение. Модули тактического уровня возвращают результат о возможности/невозможности выполнить задание по перемещению с заданными ограничениями и, при необходимости, процедура перепланирования на стратегическом уровне вносит согласованные с остальными участниками коалиции корректизы в общий план.

На тактическом уровне управления решаются задачи навигации объекта управления в пространстве, основными из которых являются: картирование (построение, обновление, уточнение модели (карты) местности), локализация (привязка объекта управления к имеющейся карте) и планирование траектории. Последняя задача разбивается на 3 фазы: прогнозирование, непосредственно построение плана и мониторинг.

В модуль прогнозирования поступает информация со стратегического уровня управления о целевой области пространства и временных ограничениях на достижение этой области объектом управления, после чего осуществляется предварительный расчет необходимых параметров движения (например, скорости) для выполнения поставленного задания. Основное отличие методов прогнозирования траектории от методов собственно планирования заключается в том, что первые могут пренебречь ограничениями на динамику движения объекта управления и другими ограничениями. За счет этого прогнозирование траектории осуществляется с минимальными затратами вычислительных ресурсов.

Рассчитанные модулем прогнозирования параметры движения передаются на нижний (реактивный) уровень управления, на котором осуществляется учет ограничений на динамику движения объекта управления, в результате чего формируются геометрические ограничения на форму траектории, соблюдение которых гарантирует возможность следования по ней при определенных ограничениях на внешние возмущения. Далее планирование осуществляется с учетом этих геометрических ограничений. Результатом работы модуля планирования является построенная траектория (тогда на верхний уровень передается соответствующее сообщение), либо сигнал о том, что при заданных ограничениях

задача планирования траектории невыполнима (за отведённое время). В последнем случае на стратегический уровень передается запрос на перепланирование, т.е. на выбор другой целевой области в пространстве и/или других временных ограничений.

Таким образом, планирование траектории представляет собой итеративный процесс с обратными связями, как от верхнего, так и от нижнего уровней управления, что является существенным отличием предлагаемой архитектуры от современных аналогов. Предполагается, что использование итеративной процесса "прогнозирование – расчет геометрических ограничений – планирование траектории – планирование поведения" позволит существенно повысить качество решения задачи интеллектуального управления сложными техническими объектами, т.е. позволит решать такие задачи, которые не могут быть решены в рамках имеющихся подходов.

Основной задачей управления сложным техническим объектом на реактивном уровне является обеспечение заданных характеристик динамического объекта с помощью управляющего воздействия по принципу обратной связи. Для этого в процедуру выработки управляющего сигнала с тактического уровня поступают желаемая траектория и желаемые параметры движения (например, скорость). Уровень может работать в двух режимах: управления реальным объектом и численного моделирования процесса управления (используется для отладки и изучения свойств построенной системы).

В первом случае информация о текущих характеристиках динамического объекта (фазовые координаты) поступает от сенсоров, а управление подается на органы управления. Во втором – информация о текущих характеристиках динамического объекта поступает от процедуры интегрирования уравнений модели динамики полета, в нее же передается и текущее управляющее воздействие.

Кроме того, на реактивном уровне происходит анализ ошибки управления, т.е. определение того, на сколько фазовые координаты объекта (реально измеренные или полученные в ходе численного моделирования) отличаются от заданных. Ошибка сравнивается с допустимым порогом, результат сравнения передается на тактический уровень в модуль мониторинга выполнения плана.

Архитектура системы управления STRL имеет ряд ключевых особенностей. Во-первых, она включает в себя основанный на моделях когнитивных функций человека стратегический уровень управления. Во-вторых, она использует новый подход в задаче планирования траекторий, предполагающее активное взаимодействие с верхним и нижним уровнями управления, в том числе, вычисление геометрических ограничений на форму траектории, которые обусловлены особенностями динамической модели объекта управления. Наконец, в-третьих, применение особого вида обратной связи на тактическом уровне позволяет распространить данную архитектуру управления не только на разные виды БПЛА, но и на другие виды сложных технических объектов.

Стратегический уровень управления

Знаковый механизм нейросимвольной интеграции для формирования представления знаний, используемого когнитивным агентом, опирается на психологические теории, в которых дается не только качественное описание свойств когнитивных функций, но и приводятся структурные описания лежащих в их основе психических образований. К таким теориям относятся культурно-исторический подход Выготского-Лурии [662], теория деятельности Леонтьева [673] и модель психики Артемьевой [661]. Согласно перечисленным психологическим подходам высшие когнитивные функции осуществляются в рамках так называемой мотивированной предметной деятельности, когда объекты и процессы внешней среды (денотаты) опосредованы для субъекта (человека или активного технического устройства) специальными образованиями, называемыми знаками. Участие знака в той или иной когнитивной функции определяется его структурой, в которую входит образ, значение и личностный смысл [677]. Образная составляющая отвечает за функции воспроизведения и различения денотата в ходе деятельности. Составляющая значения представляет собой место данного знака в той или иной знаковой системе, которая отражает в функциональном смысле наработанные общей практикой коллектива-владельца данной знаковой системы способы использования опосредованного предмета или процесса. Иными словами в значении знака заключены общие знания о функционировании данного предмета или процесса. Наконец, составляющая личностного смысла несет в себе собственный опыт действий субъекта с денотатом знака, который выражается, в том числе, и в интегральной оценке роли этого денотата в его текущей деятельности: способствует ли данный процесс или предмет достижению текущего мотива.

Трехкомпонентная структура элемента картины мира, которая, как было сказано выше, в психологии называется знаком, подтверждается и рядом работ нейрофизиологов [665; 690]. Кроме того, по современным нейрофизиологическим представлениям строение коры головного мозга практически однородно во всем своем объеме (наличие колонок неокортекса). При этом множество связей между достаточно малыми зонами коры (так называемый коннектом [559]) явно указывают на иерархичность ее строения и на присутствие как восходящих, так и обратных, нисходящих связей. Отсюда следует, что компоненты элемента картины мира должны обладать иерархическим однородным строением с восходящими потоками информации и нисходящей обратной связью. Кроме того, образная компонента должна иметь такую функцию распознавания, которая при категоризации статических объектов и динамических процессов использует обратную связь для предсказания сигнала в следующие моменты времени.

Знаковая картина мира в архитектуре STRL выполняет роль одной из реализаций нейросимвольной интеграции, которая необходимо должна присутствовать в системах управления общего назначения предполагающих символическую коммуникацию в коалиции или с оператором.

Для формального описания структуры элемента картины мира была предложена [81] иерархия следующих автоматов Мили с переменной структурой и конечной памятью (распознающий автомат или R -автомат):

$$R_i^j = \left\langle X_i^j \times \hat{X}_i^{j+1}, 2^{Z_i^j}, X_i^{*j} \times \hat{X}_i^j, \varphi_i^j, \vec{\eta}_i^j \right\rangle$$

где i - сквозной индекс автомата в иерархии, j - уровень иерархии, X_i^j - множество входных сигналов, X_i^{*j} - множество выходных сигналов, \hat{X}_i^{j+1} - множество управляющих (предсказывающих) сигналов с верхнего уровня иерархии, \hat{X}_i^j - множество управляющих (предсказывающих) сигналов на нижний уровень иерархии, $2^{Z_i^j}$ - множество состояний (множество подмножеств множества матриц предсказания (см. далее)), $\varphi_i^j : X_i^j \times \hat{X}_i^{j+1} \rightarrow 2^{Z_i^j}$ - функция переходов, $\vec{\eta}_i^j : 2^{Z_i^j} \rightarrow X_i^{*j} \times \hat{X}_i^j$ - вектор-функция выходов. Входные, выходные и управляющие сигналы представляют собой вектора действительных чисел, каждый компонент которых является весом распознаваемого или входного признака.

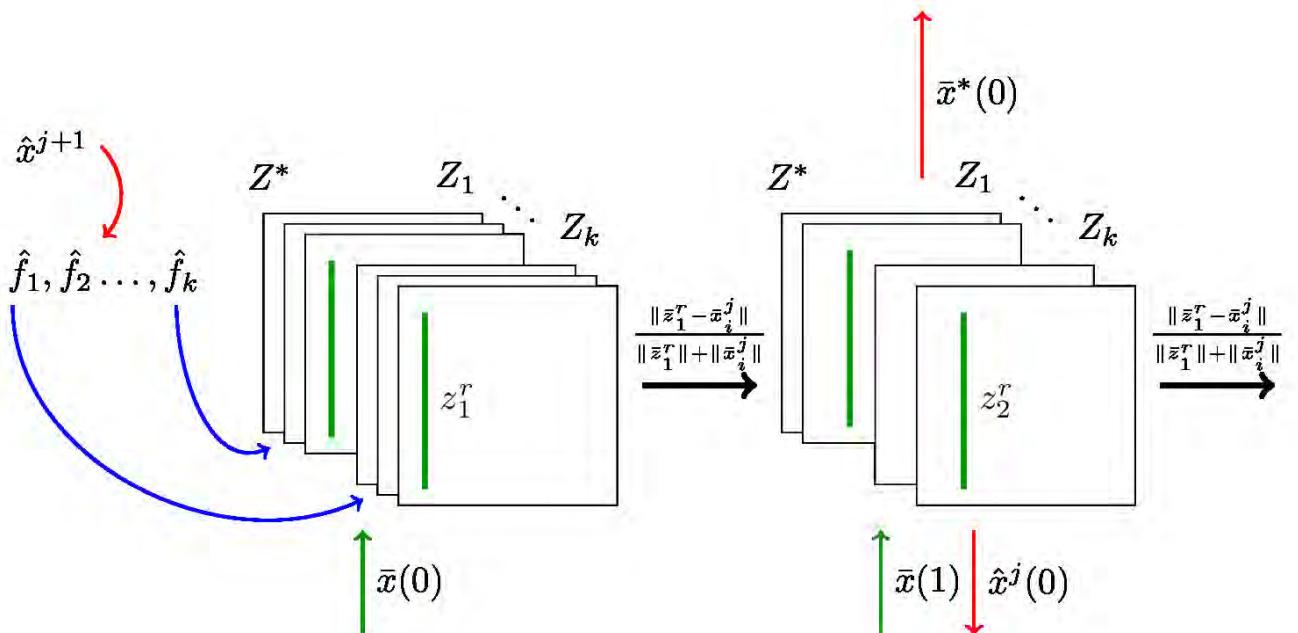


Рисунок 2.4 — Схема работы распознающего автомата (нижние индексы опущены). Из множества функций распознавания \hat{F}_i^j по управляющему вектору \hat{x}^{j+1} , поступающему с верхнего уровня иерархии, выбирается подмножество активных функций, по которому формируется множество активных матриц предсказания Z^* . Это множество фильтруется в каждый момент времени в соответствии с декартовым расстоянием до входного вектора x^j . Также каждый момент времени вычисляются выходной вектор \bar{x}^* и управляющий вектор \hat{x}^j , отправляемый на нижний уровень иерархии.

В качестве функции распознавания k -ого выходного признака \hat{f}_k в R -автомате удобно использовать набор булевых матриц предсказания $Z_k = \{Z_1^k, Z_2^k, \dots, Z_m^k\}$, в которых каждый столбец \bar{z}_u^r является вектором предсказания входных признаков в момент времени $\tau_s + u$, где τ_s – начало вычислительного цикла (момент действия управляющего сигнала \hat{x}_i^{j+1}). Сама матрица Z_r^k задаёт последовательность битовых векторов, наличие 1 в котором свидетельствует о необходимости соответствующего входного признака для распознаваемого

функцией \hat{f}_k признака. Алгоритм \mathcal{A}_{th} вычисления функции переходов φ_i^j и выходной функции $\vec{\eta}_i^j$ по начальному моменту времени τ_s , управляющему воздействию $\hat{x}_i^{j+1}(\tau_s)$ и входному воздействию ω_i^j представлен на рисунке 2.4.

Введение такого автомата и некоторых соотношений на множестве R -автоматов позволяет определить все компоненты знака. Так, образом знака s , соответствующего признаку f_1 , будет называться такое подмножество $p(s)$ признаков, что $\forall f_i \in p(s) f_i \sqsubset f_1$. Здесь отношение \sqsubset является отношением распознавания одного признака с помощью другого (выход распознающего автомата для признака f_i является входом распознающего автомата для признака f_1). Выделение подмножества процедурных признаков, опосредующих действия и процессы, в которых столбцы матрицы предсказания разделены на столбцы условий и столбцы эффектов, ведет к следующему определению значения. Если f_1 – признак, соответствующий знаку s , f_2 – процедурный признак, $f_1 \sqsubset^c f_2$, то будем называть f_2 элементом значения признака f_1 . Отношение \sqsubset^c показывает, что признак f_2 находится в столбце условий матрицы предсказания признака f_1 . Наконец, введение выделенного подмножества личностных признаков F_I приводит к определению личностного смысла: если f_1 – признак, соответствующий знаку s , f_2 – процедурный признак, $f_1 \sqsubset^c f_2, \exists f_I \in F_I f_I \sqsubset^c f_2$, то будем называть f_2 элементом личностного смысла признака f_1 .

На множестве компонент знака в процессе деятельности субъекта возникают определенные отношения, что, в свою очередь, приводит к формированию картины мира. Для описания ситуаций в картине мира используются три типа семантических сетей: сеть на образах знаков, сеть на значениях и сеть на личностных смыслах. Процессы самоорганизации на этих сетях включают в себя помимо пополнения семейств отношений также и образование новых узлов сети, что соответствует образованию нового элемента картины мира.

Процесс формирования нового знака включает в себя определение связей между компонентами знака, а затем именование получающейся структуры [677]. На первом этапе происходит итерационный процесс формирования будущего образа знака на основе поступающей из внешней среды эталонного значения (например, из сообщения от другого участника коалиции). Данный итерационный процесс в случае его сходимости приводит к связыванию образа, значения и имени знака. Личностный смысл формируется на основе прецедентов действия с опосредуемым предметом или процессом.

Рассмотрение процедурных признаков в виде правил с определенными множествами добавляемых и удаляемых признаков позволяет сформировать алгоритм основного итерационного процесса образования знака с помощью R -автоматов [43].

Изложенное выше представление знаний, используемое на стратегическом уровне управления БПЛА, позволяет описывать высокоуровневые ситуации, представляющие собой множество знаков, с некоторыми отношениями на именах этих знаков, транслируемые с одного из трех типов семантических сетей на компонентах. Такое описание ситуаций включает как пространственно-временные данные, так и данные о поведении других участников коалиции [83]. Сообщения, получаемые системой управления от других участников коалиции, представляют собой формальные записи ситуаций, т.е. перечисление

имен знаков и типов отношений, связывающих эти имена. Общая часть картины мира участников коалиции включает в себя имена знаков и компоненты значений, что обеспечивает корректное прочтение сообщения адресатом, в том случае, если сообщения не будут содержать противоречащую образную и смысловую составляющие.

Знаковое представление знаний позволяет построить алгоритмы планирования поведения, целеполагания и распределения ролей в коллективе БПЛА. Так, алгоритм планирования поведения, учитывающий действия других участников коалиции, представляет собой поэтапное построение цепочки прогнозируемых ситуаций на основе компонент знаков, составляющих предыдущие в цепочке ситуации (в том числе начальные и целевые ситуации, которые являются входными данными для алгоритма планирования). Для построенной ситуации конструируется действие (личностный смысл ситуации), которое оценивается как успешное, приближающее к целевой ситуации. План, таким образом, представляет собой последовательность действий, переводящих начальную, текущую ситуацию в конечную, целевую. Действием – этапом плана – может быть также и действие другого участника коалиции, если соответствующий ему знак с необходимыми компонентами присутствует в картине мира. Актуальность знаков, опосредующих других участников коалиции, поддерживается за счет обмена сообщениями, что также обеспечивает и согласование общего плана выполнения задачи. Подробно алгоритмы планирования и распределения ролей в коалиции интеллектуальных агентов описаны в работе [43].

После составления плана, запускается процесс его реализации: из описания промежуточных ситуаций выделяется пространственно-временная составляющая и передается на нижний тактический уровень управления. В случае успешного построения и следования по траектории происходит переход к следующей промежуточной ситуации. В случае же возникновения препятствий (невозможно построить или пролететь по траектории с заданным временным ограничением) поступает сигнал к перепланированию и происходит новая оценка ситуации и обновление плана.

Тактический уровень управления

На тактическом уровне управления решаются задачи навигации БПЛА в окружающей среде. При этом модули уровня взаимодействуют как с модулями стратегического, так и с модулями реактивного уровней, поэтому тактический уровень помимо реализации навигационного функционала является связующим уровнем STRL-архитектуры. Предполагается, что функциональные процессы, протекающие на верхнем и нижнем уровнях управления, допускают пространственную интерпретацию части операционных (рабочих) данных, что обуславливает возможность интеграции этих данных в пространственную модель мира (ПММ), которая является базой знаний тактического уровня управления, и за счет чего и осуществляется межуровневая связь. Другими словами, предполагается, что с верхнего уровня иерархии поступают сигналы (соответствующие определенным

высокоуровневым целям) о том, какого пространственного положения (и за какое время) необходимо достичь объекту управления и эти сигналы могут быть встроены в пространственную модель мира на тактическом уровне. В то же время реактивный уровень реализует (посредством отдельного модуля) функционал преобразования ограничений на динамику движения объекта управления в пространственные ограничения (геометрические, метрические и др.), которые также допускают интерпретацию в рамках используемой пространственной модели. К основным задачам тактического уровня управления относятся: определение текущего состояния БПЛА относительно пространственной модели мира (фактически – локализация БПЛА), поддержание модели в актуальном, непротиворечивом состоянии и осуществление планирования в рамках этой модели (планирование траектории).

В современной робототехнике и искусственном интеллекте первые две задачи обычно объединяются в одну – задачу одновременного картирования и локализации (*simultaneous localization and mapping, SLAM*). Методы решения этой задачи существенно зависят от того, какими входными данными располагает объект управления, что, в свою очередь, зависит от того, какими датчиками он оснащён. Если у БПЛА имеется лазерный дальномер, то для локализации и картирования обычно применяются алгоритмы сопоставления сканов (*scan matching*), хорошо зарекомендовавшие себя в области наземной робототехники (например, [288]). Если БПЛА оснащён стереопарой, то используются методы вычислительной геометрии и компьютерного зрения [640]. Если же БПЛА оснащён одной камерой, то возможно применение следующих, активно разрабатываемых в последнее время методов: LSD-SLAM [251], PTAM [355], ORB-SLAM [455] и др. Необходимо отметить, что во многих случаях в качестве входных данных для решения задачи одновременного картирования и локализации используется также пространственная модель (карта) известная априори (возможно содержащая неточности). Также стоит упомянуть, что в некоторых случаях, задача локализации становится тривиальной (например, когда осуществляется полет в открытой местности, а БПЛА оснащен датчиками приема сигналов глобальных позиционных систем (GPS, ГЛОНАСС)). Подводя итог вышесказанному, отметим, что в рамках STRL-архитектуры не предлагается использование конкретных методов картирования и локализации, т.к. указанные методы существенным образом зависят от модели конкретного БПЛА, для которого разрабатывается система управления.

Методы планирования траектории, в отличие от вышеперечисленных методов одновременного картирования и локализации, являются более универсальными, т.к. обработка непосредственно сенсорной информации при планировании не производится. В рамках STRL-архитектуры предполагается, что в качестве входных данных для метода построения траектории выступают пространственная модель местности, а также начальное и целевое положение объекта управления в этой модели. Обычно ПММ это граф, вершинам которого соответствуют координаты возможных положений БПЛА (центра масс) в пространстве, а ребрам – т.н. элементарные траектории, т.е. такие траектории следование по которым обеспечивается реактивным уровнем управления в автоматическом режиме. Следовательно, задача планирования траектории сводится к задаче поиска пути на графике.

В рамках STRL-архитектуры предлагается использование графов особой структуры, а именно – графов регулярной декомпозиции, метрических-топологических графов (МТ-графов) [691] – в качестве пространственной модели местности. Эти графы являются простыми, и вместе с тем информативными моделями, активно используемыми в робототехнике [244; 261]. Для поиска пути на МТ-графе (графе регулярной декомпозиции) в рамках STRL-архитектуры предлагается использовать авторский метод поиска пути – LIAN [692], который позволяет учитывать ограничения на форму траектории, формируемые на реактивном уровне управления. Результатом планирования при таком подходе является траектория, следование вдоль которой гарантировано обеспечивается модулями реактивного уровня управления при определенных ограничениях на внешние возмущения (траектория, учитывающая ограничения на динамику полета БПЛА). Заметим, что альтернативные подходы к учету ограничений на динамику движения объекта управления [364; 521] приводят к существенному росту пространства состояний, что заметно снижает их эффективность при решении практических задач.

Предлагается следующая модель геометрических ограничений. Считается, что реализуемая траектория представляет собой последовательность отрезков таких, что угол отклонения между любыми смежными отрезками последовательности не превышает (по модулю) некоторого фиксированного значения α . Для определения таких ограничений предложен метод, основанный на анализе области достижимости динамических систем. Для построения траектории, учитывающей эти ограничения, предложен метод LIAN [692]. Этот метод, может рассматриваться как модификация классического алгоритма эвристического поиска – A^* [311]. Поиск осуществляется в пространстве особой структуры, при этом используются дополнительные правила отсечения. Другими косвенно близкими к LIAN алгоритмами являются R^* [409] и $Theta^*$ [617]. Подчеркнем, однако, что ни один из упомянутых алгоритмов не позволяет учитывать описанные выше ограничения на форму траектории (т.е. неприменим в рамках STRL-архитектуры). Подробно алгоритм и его свойства рассмотрен в работах [651; 692]. Пример построенной алгоритмом траектории приведен на рисунке 2.5.



Рисунок 2.5 – Траектория, построенная алгоритмом LIAN.

Для формирования пространственных ограничений на форму траектории модулям реактивного уровня должны быть переданы в качестве входных данных, определяющие желаемые параметры движения объекта управления (например, скорость БПЛА). Для получения этих данных используется модуль прогнозирования траектории. Фактически прогнозирование траектории осуществляется за счет быстрого планирования траектории,

без учета ограничений (пространственных (препятствия), ограничений на динамику движения и др.), для чего могут использоваться вычислительно эффективные модификации эвристического поиска, такие как, например, алгоритм JPS [309]. В простейшем случае прогнозирование может быть сведено к построению отрезка прямой, соединяющей начальное и целевое положение. Зная длину этого отрезка, очевидно, можно вычислить значение минимальной требуемой скорости движения, которое и подлежит передаче на реактивный уровень управления для формирования геометрических ограничений (вычисления значения угла α).

Оставшимся неописанным модулем тактического уровня управления является модуль мониторинга выполнения плана (т.е. следования по траектории). Этот модуль получает информацию (от модулей нижестоящего уровня) о текущем положении БПЛА и сравнивает это положение с имеющимся планом (построенной траекторией). Если отклонение превышает заданное (или вычисляемое) пороговое значение, то цикл планирования («прогнозирование» - «вычисление геометрических ограничений» - «планирование с геометрическими ограничениями») повторяется. Если из текущего положения не удается построить траекторию до целевой области, то соответствующее сообщение отправляется на стратегический уровень управления и инициируется процесс смены цели.

Реактивный уровень управления

Основной задачей на реактивном уровне является обеспечение заданных характеристик динамического объекта (желаемой скорости, местоположения и др.), полученных с тактического уровня. Это осуществляется с помощью процедуры выработки управляющего сигнала и анализа динамики. В случае режима управления реальным объектом, управляющий сигнал подается на органы управления БПЛА: двигатели винтов, элероны и т.д., а в режиме моделирования – в процедуру интегрирования уравнений модели динамики полета. Так или иначе, для выработки управляющего сигнала необходимо решить задачу построения регулятора. Другой задачей реактивного уровня является определение геометрических ограничений на допустимую траекторию перемещения при заданных тактическим уровнем параметрах полета. Для ее решения разработана специальная процедура.

Далее опишем решение задач построения регулятора и определение ограничений, предлагаемое в рамках построенной архитектуры.

В области автоматического управления существует множество подходов и техник к построению регулятора. Некоторые из них косвенно связаны с когнитивной деятельностью человека на уровне реализации и контроля моторных действий. Так, например, управление на основе искусственных нейронных сетей [164], использует принцип обучения по примерам, а управление на основе нечеткой логики [271] использует формализованное интуитивное знание экспертов предметной области. Применение этих техник особенно актуально,

когда нет возможности построить адекватную математическую модель БПЛА. Основным недостатком этих подходов является отсутствие строгого доказательства работоспособности созданных регуляторов.

С другой стороны, существует множество свободных от этого недостатка строгих с математической точки зрения техник, не опирающихся на биологическую моторную деятельность, например, линейно-квадратичный синтез, H_{∞} синтез, линеаризация обратной связью, бекстеппинг (backstepping) и др. К их недостаткам относится ограниченная область применения: они работают либо с линейной моделью БПЛА, либо с нелинейной моделью особого вида.

В описываемой архитектуре STRL используется один из перспективных подходов, основанных на уравнении Риккати с зависящими от состояния коэффициентами (State Depended Riccati Equation, SDRE). Работы в этой области активно ведутся с середины 90-х годов прошлого века [177; 448]. Развитие и применение этого подхода, с одной стороны, позволяет получить достаточно общую методологию для построения субоптимальных гладких нелинейных регуляторов для нелинейных систем, коэффициенты которых также зависят от состояния. С другой стороны, в SDRE управлении можно использовать некоторые принципы, по которым реализуются низшие когнитивные функции человека, связанные с контролем и выполнением моторных действий. А именно: адаптировать работу регулятора в зависимости от режима и условий работы и существующих ограничений на управление. Это достигается с помощью задания линейного квадратичного функционала качества, отражающего требования к качеству переходного процесса и содержащего две весовые матрицы: при состоянии системы и при управлении. Элементы этих весовых матриц являются нелинейными гладкими функциями состояния, что позволяет задавать различные требования к переходному процессу в зависимости от режима работы системы (областей фазового пространства), а также учитывать существующие ограничения на управление. Например, при посадке БПЛА необходимо повышать требования к точности траектории, т.е. увеличивать значения соответствующих весовых элементов в матрице при состоянии, и, следовательно, изменять закон управления. Другой пример – создание с помощью весовой матрицы при управлении области в фазовом пространстве, где коэффициент усиления должен быть понижен, т.к. есть вероятность выхода управления «на упор». Регулятор стремится минимизировать значение функционала качества. Для этого при каждом значении вектора состояния численно решается соответствующее матричное уравнение Риккати, учитывающее как изменение весовых матриц в критерии качества, так и нелинейности самого БПЛА.

Синтезированные этим способом регуляторы могут достаточно хорошо компенсировать нелинейности управляемой системы, учитывают [182; 183] ограничения на управление и состояние системы, способны непрерывно изменять алгоритм работы регулятора в зависимости от сценария задачи. Однако, важным вопросом в SDRE управлении является компромисс между точностью решения и вычислительной сложностью метода.

В рамках архитектуры STRL предложен новый подход к построению SDRE нелинейного регулятора в численно-аналитической форме. Традиционные подходы в

области SDRE управления связаны с необходимостью применения поточечного вычисления уравнения Риккати, интерполяционных процедур на полученной сетке, а также проведения различных символьных вычислений, что в реальных задачах может потребовать больших вычислительных ресурсов. Применение аналитических формул существенно снижает вычислительную сложность выработки управления с помощью SDRE техники [664]. Ниже кратко описан предложенный подход.

Пусть имеется нелинейная система управления моторными действиями на полуоси с линейно входящим управлением, где коэффициенты могут быть зависимыми от состояния [682]:

$$\frac{dx}{dt} = (A_0 + A_1(x))x + (B_0 + B_1(x))u, \quad x(0) = x^0, \quad x \in X \subset R^n, \quad u \in R^r, \quad t \in [0, \infty), \quad (2.1)$$

где $A_0, A_1(x) \in R^{x \times n}, B_0, B_1(x) \in R^{n \times r}, X$ – некоторое ограниченное множество пространства состояний БПЛА (рабочие режимы), u – управление, x – вектор состояния (вектор фазовых координат БПЛА). Для любого непрерывного управления траектория системы 2.1 существует и единственна на $[0, \infty)$.

Формулируется следующая задача управления. Требуется построить такое управление $u(x)$, что траектория замкнутой системы 2.1 будет асимптотически стремиться к нулю, т.е. $x(t, \varepsilon) \rightarrow 0$ при $t \rightarrow \infty$ и

$$I = \frac{1}{2} \int_0^\infty (x^\top Q(x)x + u^\top R(x)u) dt \rightarrow \min, \quad (2.2)$$

где I – функционал качества, который минимизируется управлением u , $Q(x), R(x)$ – заданные положительно полуопределенная и положительно определенная весовые матрицы, определяющие «значимость» ошибок управления и расхода управления соответственно. Задавая различные зависимости элементов этих матриц от вектора состояния можно изменять алгоритм работы регулятора в зависимости от режима работы БПЛА. Далее для простоты изложения будем считать матрицу $R(x) = R_0$ постоянной.

Представление объекта управления в виде 2.1 обладает рядом преимуществ. Оно позволяет:

- отражать достаточно широкий спектр нелинейности реальной системы;
- позволяет строить управление на основе номинальной модели системы;
- выполнять строгое обоснование предлагаемого подхода на основе второго метода Ляпунова.

Некоторые общепринятые для SDRE техники условия выглядят следующим образом:

- элементы матриц $A_1(x), B_1(x)$ и их частные производные есть гладкие функции и равномерно ограничены в некоторой рабочей области $G = \{x \in X, t \geq 0\}$;
- пара матриц A_0, B_0 управляемы, а пара матриц $A_0, \sqrt{Q_0}$ наблюдаемы.

Если эти условия выполняются, то решение задачи 2.2 в рамках SDRE подхода имеет вид:

$$u(x) = -R_0^{-1} B^\top(x) K(x) x, \quad (2.3)$$

где $K(x)$ – положительно определенное решение соответствующего уравнения Риккати для текущего значения $x(t)$. Поскольку объект динамический, т.е. x может непрерывно меняться, то для выработки управления 2.3 необходимо решать уравнение Риккати в каждый момент времени. Последнее технически может быть реализовано лишь с некоторым времененным шагом, обусловленным вычислительной сложностью решения конкретной задачи.

В архитектуре STRL используется подход [663], согласно которому управление $u(x)$ ищется в виде

$$u(x) = u_0 + u_1(x), \quad (2.4)$$

где u_0 – управление 2.3 для номинальной системы 2.1 (т.е. в которой $A_1(x) \equiv 0, B_1(x) \equiv 0$) при номинальном значении функционала (т.е. когда $Q(x) = Q_0$ – известная постоянная матрица). Управление u_0 вычисляется лишь один раз. Дополнительный нелинейный член $u_1(x)$ вычисляется путем интегрирования специальной матрицы, представленной в аналитической форме. Таким образом, отсутствует необходимость решения матричного уравнения Риккати для каждого $x(t)$. Вместо этого мы имеем дело с известным аналитическим выражением, решение которого может быть оценено заранее. Следовательно, существенно снижаются вычислительные затраты на выработку управления, что особенно актуально для бортовых вычислителей или управления БПЛА с быстрой динамикой. Отметим, что управление 2.4 не является оптимальным для задачи 2.2, и имеет некоторый субоптимальный характер.

С помощью функции Ляпунова $V(x,t) = x^\top(K(x))x > 0, \forall x \neq 0$ при некоторых дополнительных условиях можно показать, что положение равновесия $x = 0$ в замкнутой системе 2.1-2.4 асимптотически устойчиво по Ляпунову, т.е. $x(t) \rightarrow 0$ при $t \rightarrow \infty$.

Таким образом, предложен численно-аналитический алгоритм построения регулятора, нелинейно зависящего от переменных состояния, что резко уменьшает объем вычислений по сравнению с применением традиционных SDRE процедур. Другой отличительной чертой регулятора является достаточная математическая строгость, а также возможность использовать некоторые принципы, на которых основывается управление моторными действиями человека: адаптировать работу регулятора в зависимости от режима, условий работы и существующих ограничений на управление.

При полете в горизонтальной плоскости тактический уровень пытается построить траекторию в виде последовательности отрезков таких, что угол отклонения между любыми смежными отрезками не превышает по модулю некоторого фиксированного значения α . Предполагается, что соблюдение этого условия гарантирует реализуемость полученной траектории, т.е. возможность выработки допустимых управляющих сигналов для следования по ней с заданной ошибкой. Предложенная процедура определения геометрических ограничений на максимально допустимый угол α основывается на анализе области достижимости динамической системы. Точное решение этой задачи, как правило, требует больших вычислительных затрат. Однако, авторами был предложен упрощенный подход, основанный на некоторых правдоподобных допущениях и обладающий достаточной общностью применения.

Предположим, что БПЛА должен следовать по траектории tr . Основное допущение метода состоит в том, что при имеющихся ограничениях на расход управления и при заданных условиях полета существует некоторая окрестность прямолинейного участка траектории tr , заданная радиусом r_d (рисунок 2.6), внутри которой всегда имеется возможность выработки допустимого управления, при котором БПЛА не покинет эту окрестность. Фактически r_d определяет «трубку», внутри которой должен происходить полет вдоль прямолинейной участка траектории tr .

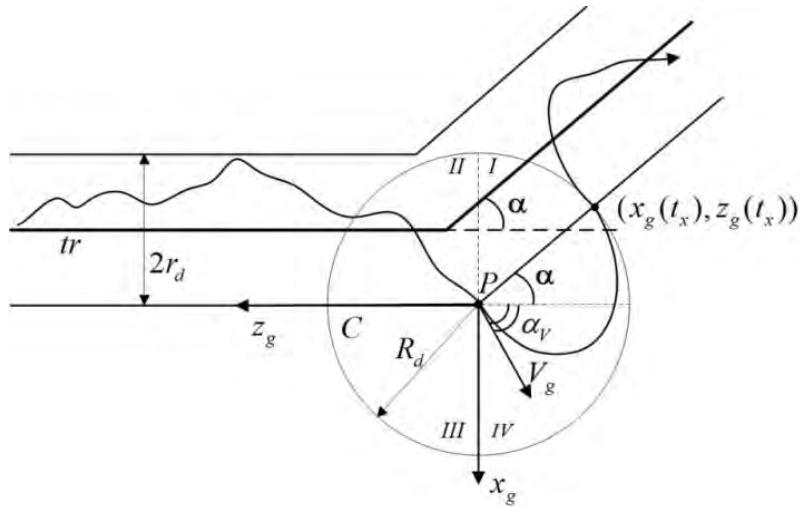


Рисунок 2.6 — К методу определения геометрических ограничений.

В методе рассматривается наихудший для управления случай при «изломе» траектории – нахождение БПЛА в точке P с максимальным углом отклонения вектора земной скорости V_g равным α_V . Выделяется свободная от препятствий область, заданная окружностью C с центром в P и радиусом R_d . Фактически, R_d определяет область совершения маневра поворота.

Задача определения угла α формулируется следующим образом. Необходимо определить такой максимальный угол α , что начав движение в точке P со скоростью V_g и используя максимальное допустимое управление БПЛА, не покидая окружность, вновь вернется в область, определяемую радиусом r_d . Тогда, согласно основному предположению, БПЛА уже не покинет допустимую область полета, заданную r_d , и продолжит следование по текущему прямолинейному участку траектории tr .

Предложенная для решения этой задачи процедура основывается на численном моделировании динамики полета при максимальном управлении, стремящемся вернуть БПЛА в область полета, определяемую радиусом r_d , и геометрическом анализе полученной траектории. Процедура была опробована для математической модели квадрокоптера AsTec Hummingbird⁶. Максимальное значение угла α в 24.4° было вычислено для параметров $V_g = 7 \text{ м/с}$, $\alpha_v = 45^\circ$, $R_d = 2.7 \text{ м}$.

Таким образом, предложен достаточно общий метод, позволяющий на основе модели динамики полета и допустимого управления, а также при некоторых разумных

⁶<http://www.asctec.de/en/uav-uas-drone-products/asctec-hummingbird/>

предположениях определить ограничения на геометрию траектории. Метод может быть адаптирован для учета внешний возмущений, например, ветра. Более полно метод изложен в [692].

2.2.2 Расширенная версия архитектуры STRL

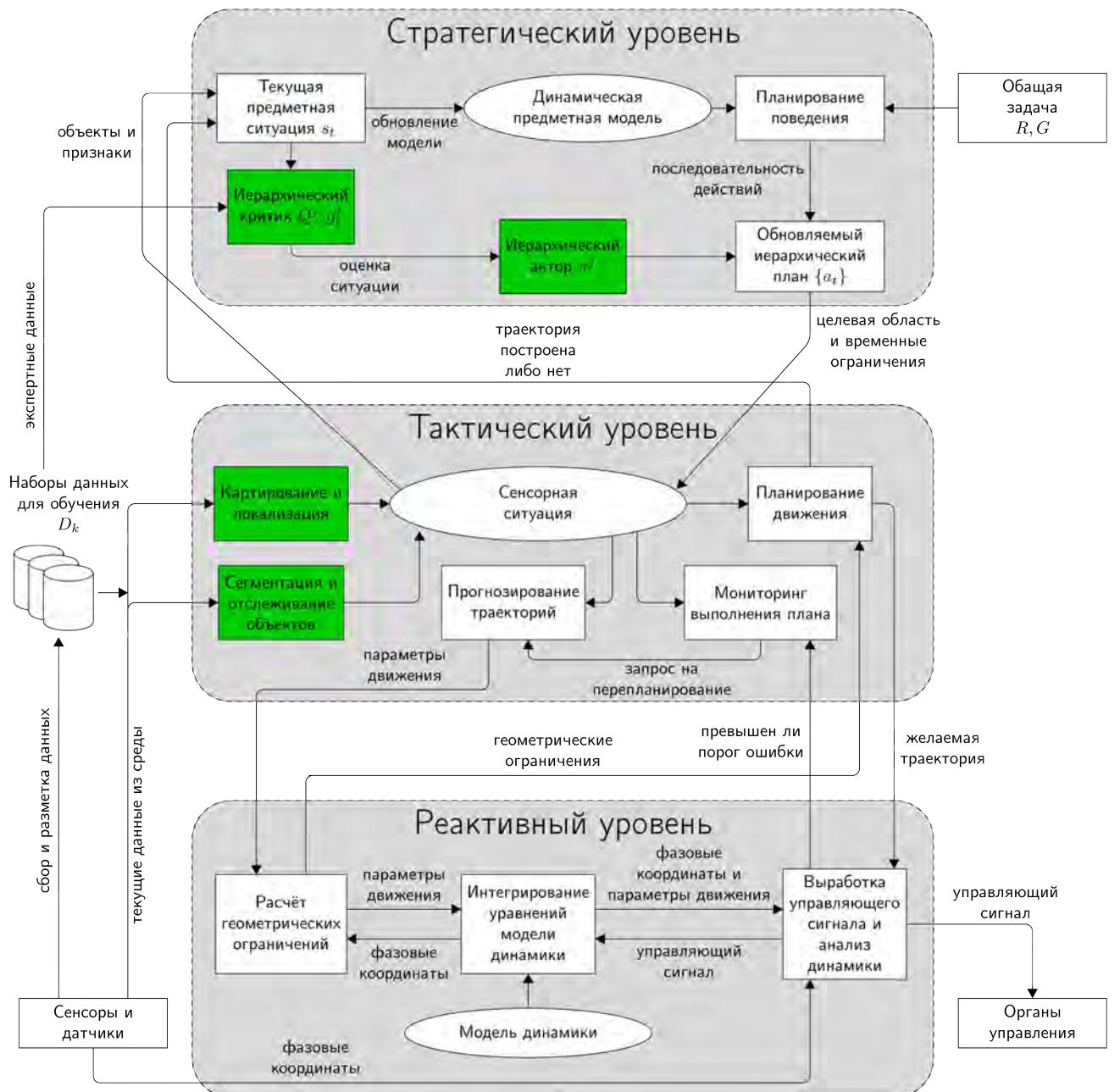


Рисунок 2.7 — Основные компоненты и межмодульное взаимодействие в архитектуре STRL2

В условиях динамической среды, свойства и поведение которой заранее не известны когнитивному агенту, в качестве которого будет пониматься мобильная робототехническая платформа, в диссертационном исследовании 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 рисунке 2.7 представлена схема основных подсистем архитектуры и базовые процедуры управления и передачи информации между модулями. Ниже кратко рассмотрены основные особенности этой архитектуры.

Архитектура STRL2 является иерархической и состоит из тех же трех базовых уровней, что и исходная архитектура STRL. На нижнем 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 [191]. Результатом является выработка управляющего сигнала на органы управления.

На тактическом уровне в STRL2 предлагается уделить больше внимания задачам компьютерного зрения, а не только построения и прогнозирования траектории движения во внешней среде. На рисунке 2.7 зеленым цветом 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 структуры), необходимых для построения пути и мониторинга движения по траектории [74].

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 [31]. Однако, и в архитектуре STRL, и в архитектуре STRL2 стратегический уровень оказывается не достаточно проработанным. Например, не рассматривается решение актуальной задачи выполнения инструкций на естественном языке. В связи с этим возникает необходимость уточнения принципов

нейросимвольной интеграции, которые должны быть реализованы на стратегическом уровне. Далее рассматривается подробная структура стратегического уровня в концепции нейросимвольной интеграции для обучения и планирования (NSLP), которая затем будет реализована для решения разного класса задач. Основными подходами к решению проблемы привязки символов являются слотовый (см. раздел 4.1) и знаковый (см. раздел 6.1) подходы.

2.3 NSLP: нейросимвольная архитектура для планирования и обучения

2.3.1 Класс задач, решаемых с использованием архитектуры NSLP

Опишем наиболее общий класс задач, для решения которого предназначена описываемая в данном разделе диссертационного исследования архитектура NSLP (neuro-symbolic learning and planning, нейросимвольное обучение и планирование). Как уже было подробно изложено в главе 1, в данной работе рассматривается ситуация последовательного принятия решений когнитивным воплощенным агентом, действующим в неизвестной внешней среде, в условиях неопределенности. Используя введенный в разделе 1.1 формализм марковского процесса принятия решений, предполагается, что агенту известно только доступное множество действий A и пространство наблюдений O . Ни функция переходов T , ни множество информационных состояний S МППР для агента не доступны. Поведение, характеризуемое последовательностью решений агента, является целенаправленным, то есть агенту предоставляется информация о множестве целевых состояний $S_g \subset S$ в следующем виде:

- через функцию вознаграждения $R = r(s_t, a_{t+1}, s_{t+1})$, сигнализирующую о которой в простейшем виде записывается как

$$r(s_t, a_{t+1}, s_{t+1}) = \begin{cases} 1, & \text{если } s_{t+1} \in S_g, \\ 0, & \text{иначе;} \end{cases}$$

- через языковую инструкцию (множество токенов) $I = \{q_1, \dots, q_n\}$, описывающую критерии достижения цели или выполнения задания.

Необходимо отметить, что задание цели не подразумевает фиксацию условий в среде, в которых будет происходить процесс принятия решений. То есть, если задано множество экземпляров среды (аналогично заданию доменов планирования в разделе 1.2) $E = \{E_1, \dots, E_m\}$, где E_i – конкретный экземпляр отличающийся от других функцией переходов T_i , соответствующего этому экземпляру МППР, то одна и та же задача $task = \langle R, I \rangle$ может быть корректно поставлена в некотором подмножестве данных сред. Целью агента $goal_i$ будет называться задача в конкретных условиях среды E_i , то есть пара $goal_i = \langle task, E_i \rangle$.

Пусть в процессе принятия решений (взаимодействия) агентом в среде E_i был получен эпизод (траектория) $\tau \sim E_i$ вида $\tau = (s_0, a_0, r_0, \dots, s_{n-1}, a_{n-1}, r_{n-1}, s_n)$. Метриками качества принятых агентом решений являются две величины:

- критерий достижения цели (success rate) SR :

$$SR(\tau, task) = \begin{cases} 1, & \text{если } s_n \in S_g(task), \\ 0, & \text{иначе;} \end{cases}$$

- мягкий критерий достижения цели (soft success rate):

$$SSR(\tau, task) = SR(\tau, task) \sum_{t=0}^{n-1} \gamma^t r_t. \quad (2.5)$$

Формулируемый класс задач будет подразумевать работу агенту в двух режимах:

Режим обучения: пусть сформулирована задача $task \langle R, I \rangle$ и имеется набор сред E , тогда задачей одновременного планирования и обучения называется задача поиска такой функции $NSLP(\theta_{enc}, \theta_{wm}, \theta_{pol}) : O \rightarrow A$ с параметрами кодировщика наблюдений θ_{enc} , модели мира θ_{wm} и стратегии θ_{pol} , что метрика SSR максимизируется на наборе сред E :

$$\max_{\theta_{enc}, \theta_{wm}, \theta_{pol}} \mathbb{E}_{\substack{NSLP(\theta_{enc}, \theta_{wm}, \theta_{pol}) \\ E_i \in E, \tau \sim E_i}} SSR(\tau, task). \quad (2.6)$$

Режим вывода (оценки): подразумевает фиксацию параметров $\theta_{enc}, \theta_{wm}, \theta_{pol}$ и запуск функции $NSLP(o_t)$ в режиме генерации действий a_t и подсчет метрики SSR по получающимся сгенерированным траекториям τ в новом наборе сред, отличающемся от используемых в режиме обучения.

Основные особенности разрабатываемой архитектуры NSLP, представленной в данной главе, заключаются в следующем (см. рисунок 2.8):

- Декомпозиция задачи последовательного принятия решений в условиях неопределенности на концептуальный (символьный) и сенсорный (субсимвольный, нейросетевой) уровни и соответственно декомпозиция функции $NSLP(o_t)$.
- Использование предобученных языковых моделей для формирования общего концептуального плана по текстовому описанию цели, то есть декомпозиция параметров θ_{wm} на два подмножества: заранее заданных параметров языковой модели мира θ_{lm} и обучаемых параметров модели мира θ_{lwm} .
- Реализация механизма одновременного обучения и планирования за счет формирования стратегии верхнего уровня для достижения концептуальной цели (подцелей) с использованием обучения с подкреплением на основе модели среды (латентной и/или объектной). Иными словами это подразумевает декомпозицию параметров стратегии θ_{pol} на два подмножества: обучаемых параметров верхнеуровневой стратегии θ_{hl} или эвристических параметров планировщика подцелей θ_{plan} .
- Реализация нейросимвольной интеграции за счет использования нейросимвольной модели для построения объектного представления сцен, то есть выбор среди класса

параметризованных с помощью набор θ_{enc} моделей таких, которые позволяют формировать объектное представление $o_t \rightarrow \{e_1, \dots, e_n\}$.

- Построение и обновление как объектной $M_O = < \hat{T}, \hat{R}, G_t >$, так и латентной модели среды $M_L = < \hat{T}, \hat{R}, enc >$, учитывающей ее динамику и действия других агентов в среде.
- Использование объектной и/или латентной модели динамики среды для осуществления планирования,
- Возможность использования навыков $\{\kappa_1, \dots, \kappa_m\}$ достижения подцелей, то есть поддержка иерархичного подхода с автоматическим выделением подцелей.
- Планирование и обучение в условиях частичной наблюдаемости.

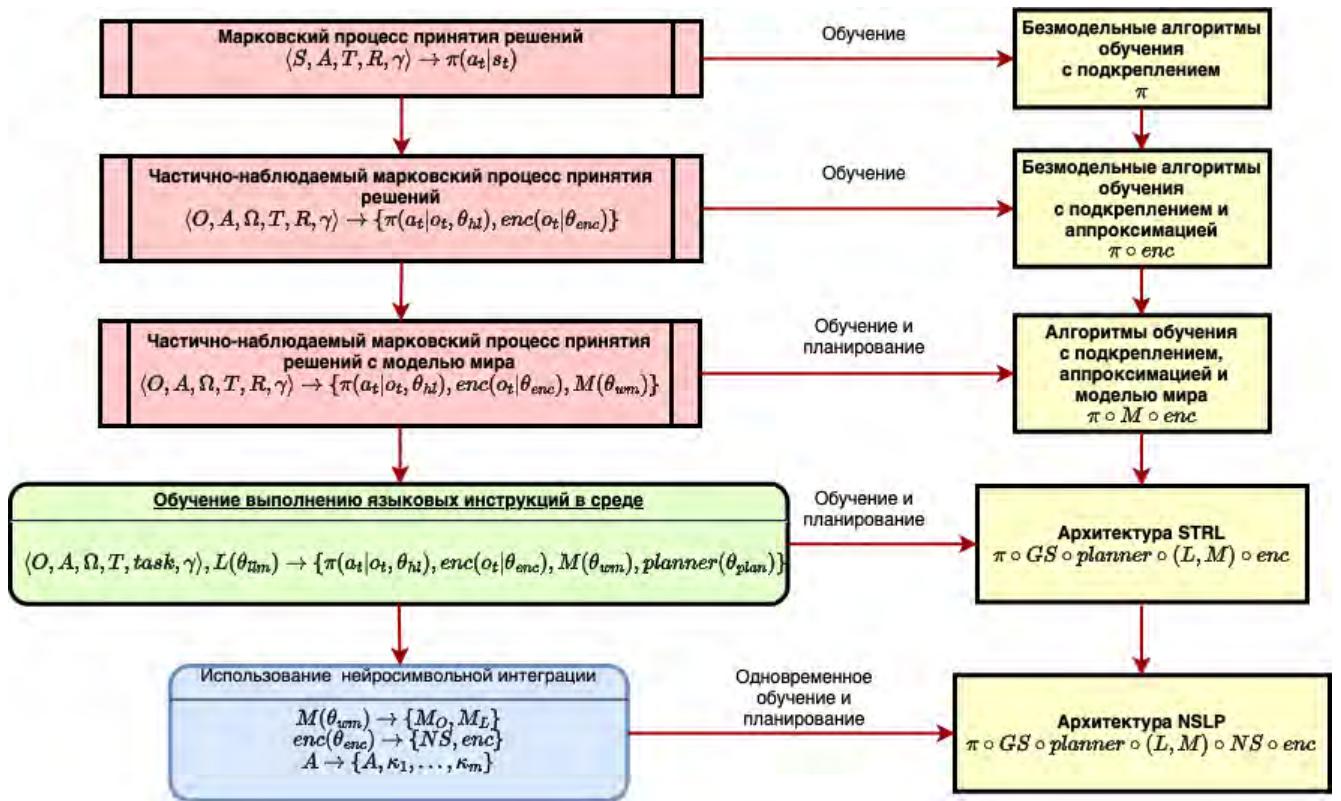


Рисунок 2.8 — Взаимосвязь ранее рассматриваемых классов задач обучения принятию решений в условиях неопределенности в динамической среде с семействами методов, предназначенных для их решения. Архитектура NSLP предназначена для решения наиболее общего класса задач обучения выполнения языковых инструкций в среде с использованием нейросимвольной интеграции для повышения качества получаемых стратегий агента.

Необходимо отметить, что за рамками данной работы находится разработка мультимодальных моделей привязки сенсорных (нейросетевых) представлений навыков и действий к концептуальным (символьным) представлениям подцелей и действий, формируемым символным планировщиком и генератором подцелей. В текущей реализации архитектуры NSLP предполагается, что такая привязка осуществляется с помощью заранее установленного фиксированного соответствия заданного набора действий и их названий и языковых обозначений. Задача разработки таких мультимодальных моделей является

логичным продолжением работ в области нейросимвольных архитектур и может опираться на такие известные мультимодальные модели как CLIP [400], R3M [517] и PVR[615].

Основной задачей предлагаемого подхода одновременного обучения и планирования является построение согласованной по иерархии модели, которая подчиняется принципу эквивалентности по полезности [616]. Ограничения, накладываемые на модель по трем составляющим: по иерархии (ограничения на возможный состав с верхнего уровня), по объектности представления (группировка признаков по классам — объектам) и по принципу соответствия текущей стратегии (эквивалентность по полезности), — способствуют формированию консистентной модели, которая позволяет строить эффективные планы, существенно ускоряющие процесс обучения. Также в данном подходе используются модели «актора-критика» с регуляризацией на функцию потерь при обучении стратегии, что позволяет проводить такое обновление модели, которое бы обеспечивало монотонность улучшения стратегии агента.

Теоретическое исследование алгоритмов обучения с подкреплением, разрабатываемых в данном разделе, представляет собой построение оценки градиента функции полезности действия $\nabla_{\theta}\hat{Q}(s,a)$ или всей стратегии $\nabla_{\omega}J(\pi)$. Такое исследование оказывается полезным в случае использования нейросетевых аппроксиматоров, для которых значение градиента определяет скорость обучения и, таким образом, эффективность алгоритма.

Основные результаты, изложенные в данном разделе, представлены в публикациях [61; 96].

2.3.2 Общая схема архитектуры NSLP

Общая схема архитектуры NSLP представлена на рисунке 2.9. Она разделена на три уровня со своим набором внешних параметров и необходимых данных для предобучения используемых компонент:

Концептуальный уровень предназначен для формирования концептуального плана с набором подцелей для достижения цели, то есть выполнения задачи $task = \langle R, I \rangle$ с функцией вознаграждения R и/или языковым описанием I в условиях конкретной среды E . Для этого используется графовое представление сцены $G_t = \langle En, Rel \rangle$, поступающее с нейросимвольного уровня, а на выходе генерируется символьный план действий $planner(\theta_{plan}) \rightarrow P$. Здесь в графе сцены $En = \{e_1, \dots, e_n\}$ — множество выделенных в среде E объектов, в том числе и другие участники деятельности (другие агенты), а Rel — множество отношений (бинарных), устанавливаемых на множестве объектов En . Внешними параметрами на этом уровне являются описание задачи $task$ и информация о других частниках совместной деятельности (например, координаты других агентов). Предобученным компонентом на данном уровне является языковая модель $L(\theta_{llm})$, которая обычно обучается в самоконтролируемом режиме на множестве

текстов и инструкций из специализированных наборов данных. Эвристические параметры Θ_{plan} планировщика *planner* также считаются заданными.

Сенсорный уровень собственно реализует одновременное обучения и планирование с использованием некоторого сенсорного представления информационного состояния s_t (в смысле марковского процесса принятия решений, формализующего взаимодействие агента и среды E), получаемого на основе сенсорной модели кодировщика $enc(o_t) \rightarrow s_t$, в свою очередь, обрабатывающей наблюдение o_t с тактического уровня STRL. Представление сцены s_t передается на нейросимвольный уровень. На выходе этого уровня при выполнении стратегии $\pi(s_t, \beta|\theta_{hl})$ и/или навыков κ_j генерируется низкоуровневое действие a_t , реализуемое конкретной операцией на тактическом или реактивном уровне STRL. β в стратегии π – это набор подцелей с нейросимвольного уровня архитектуры NSLP. Здесь в качестве внешних параметров выступает конкретная модель сенсорного ситуации enc , то есть модель кодировщика наблюдения агента o_t , сгенерированного на тактическом уровне STRL (например, в качестве o_t могут выступать представления мультимодальных или сегментационных карт). Для предобучения библиотеки навыков $\{\kappa_1, \dots, \kappa_m\}$, используемых в верхнеуровневой стратегии π , применяются симуляторы внешней среды E с заданными сценариями для предобучения конкретных навыков.

Нейросимвольный уровень собственно связывает концептуальный и сенсорный уровни за счет двух ключевых моделей: нейросимвольной объектной модели $NS(s_t|\theta_{enc})$ и мультимодальной модели привязки $MM(s_t)$, сопоставляющих названия (имена) объектов и действий с их внутренними представлениями сенсорного уровня. Результатом работы модели NS является набор представлений объектов $\{e_1, \dots, e_n\}$, которые на концептуальном уровне уже могут быть проинтерпретированы в виде графа сущностей G_t (например, с помощью графовой нейронной сети). На нейросимвольном уровне происходит разложение верхнеуровневого плана P на множество подцелей $\beta = \{\beta_1, \dots, \beta_k\}$ с помощью генератора подцелей $GP(P, s_t) \rightarrow \beta$. Предполагается, что текстовое представление этапов плана с помощью мультимодальной модели привязки MM сопоставляется с внутренними представлениями подцелей β_i . В простейшем случае такое сопоставление может реализовываться заранее заданными отображением. Внешними параметрами служат конкретная реализация объектной модели (знаковый или слотовый подходы к представлению объектов) и мультимодальные наборы данных (обычно пары текст-изображение) для предобучения как данной объектной модели NS , так и мультимодальной модели привязки MM .

Необходимо отметить, что предполагается, что NSLP встроена в более общую STRL архитектуру и наблюдения $o_t \in O$ в нее поступают с тактического уровня STRL, а действия $a_t \in A$, по сути, являются, в свою очередь, операциями, реализуемыми на тактическом и реактивном уровне за счет тактического планирования и реактивного следования конкретной траектории в обобщенном конфигурационном пространстве (например, для мобильной робототехнической платформы и манипулятора). Далее опишем функционирование архитектуры NSLP в двух режимах: вывода (inference), когда используется текущая архитектура и идет накопление опыта (памяти предшествующих действий D , данных для обучения

моделей NS, enc, M_O, M_L , и обучения (learning) стратегии π и моделей с использованием накопленного опыта.

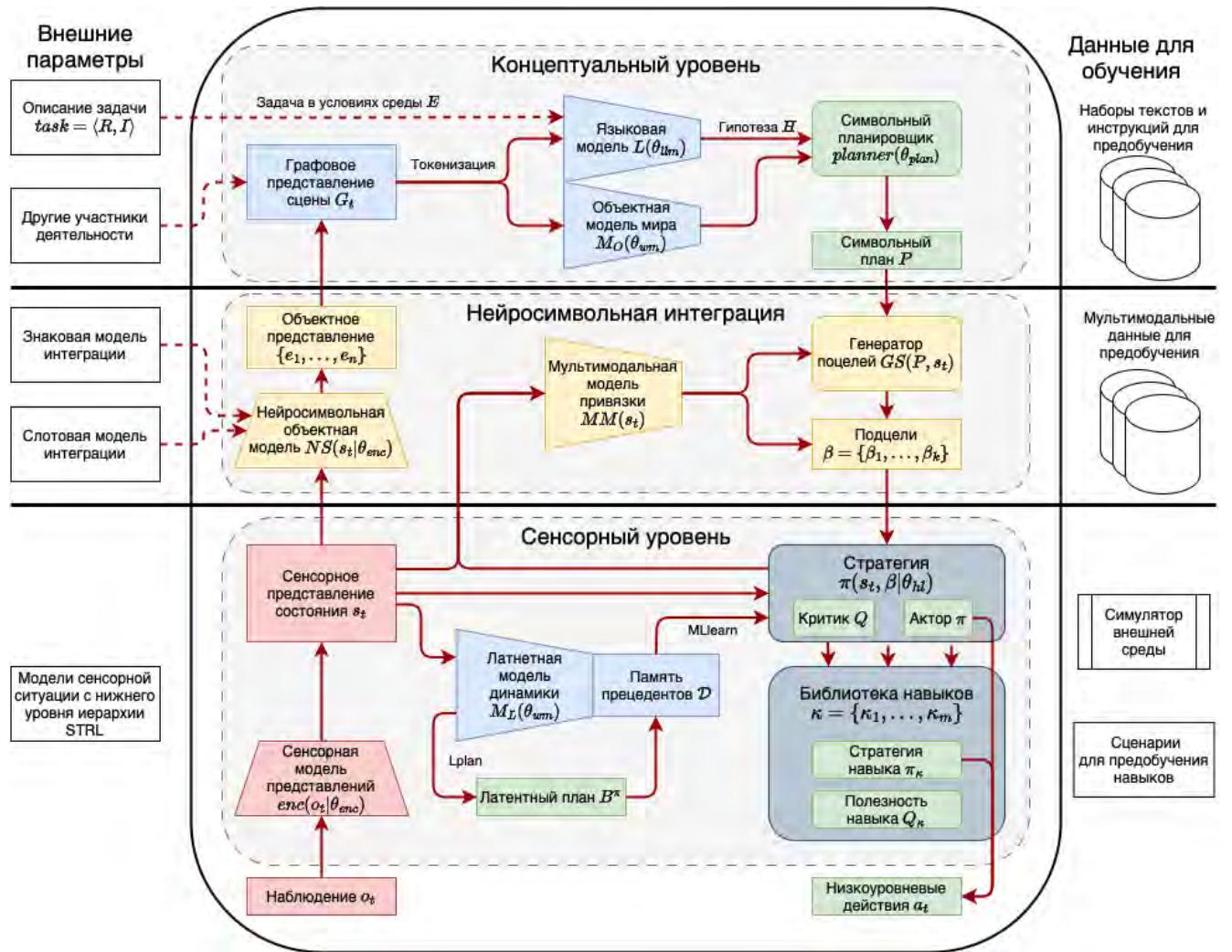


Рисунок 2.9 — Архитектура NSLP агента с подсистемами обучения и планирования поведения. Выделены три составляющих архитектуры: концептуальный уровень планирования, сенсорный уровень одновременного обучения и планирования, а также уровень нейросимвольной интеграции с объектным представлением сцен. Красным цветом помечены сенсорные компоненты, синим — компоненты модели мира (объектной и латентной), зеленым — компоненты действия (план, стратегия, полезность действий), желтым — нейросимвольные интеграционные компоненты. Прямоугольниками обозначены внутренние представления, трапециями — обучаемые модели, прямоугольниками со скругленными углами — алгоритмические процедуры обучения или принятия решений.

Режим вывода

В режиме вывода когнитивный агент на основе архитектуры NSLP фиксирует все параметры обучаемых компонент архитектуры: языковой модели $L(\theta_{lm})$, объектной модели

мира $M_O(\theta_{wm})$, нейросимвольной объектной модели $NS(\theta_{enc})$, мультимодальной модели привязки MM , сенсорной модели представлений $enc(\theta_{enc})$, латентной модели динамики $M_L(\theta_{wm})$, стратегии агента $\pi(\theta_{hl})$ и библиотеки навыков κ – и использует их в генеративном режиме, получая на входе наблюдение o_t и выдавая низкоуровневые действия a_t . По сути агент реализует многопараметрическую многокомпонентную функцию

$$NSLP(o_t) = \pi \circ GS \circ planner \circ (L, M_O) \circ NS \circ enc \rightarrow a_t. \quad (2.7)$$

В данном режиме не используются латентная модель динамики M_L , критики верхнеуровневой стратегии π и всех навыков из библиотеки κ . Весь накопленный когнитивным агентом опыт (как правило, это наблюдения o_t , действия a_t и вознаграждения r_t в каждый момент времени t) сохраняется в память прецедентов \mathcal{D} . Возможна ее полная очистка после завершения режима обучения (так называемый on-policy вариант обучения по актуальному опыту) либо сохранение всей или некоторой приоритизированной информации (off-policy обучение по отложенному опыту).

Во время взаимодействия агента со средой активно используется объектная модель мира M_O для предсказания возможных последствий плана, который генерируется символьным планировщиком $planner$. Латентная модель мира M_L используется для дополнительного пополнения памяти прецедентов при планировании в режиме «воображения» при обучении.

Режим обучения

В режиме обучения используется накопленная память прецедентов \mathcal{D} для обновления верхнеуровневой стратегии агента по достижениям конкретных подцелей β_i , полученных из символьного плана P с помощью генератора подцелей GS . В режиме «актор-критик» обновляется также функция полезности Q , которая, в соответствии с теоремой о градиенте стратегии 1.1.3, используется для коррекции градиентного шага, вычисляемого по функции полезности стратегии π . Также возможно дообучение языковой модели L на сформированных признаках успешности выполнения сгенерированного плана на основе вознаграждений из памяти прецедентов (в режиме RLHF [399] или DPO [226]).

В режиме обучения также возможно обучение (дообучение) сенсорной модели enc . Однако это возможно только в некоторых средах в связи с неустойчивостью всего процесса оптимизация такой многокомпонентной функции как $NSLP$. В упрощенных версиях архитектуры обучение кодировщика является полноценным этапом обучения всей стратегии π и по сути означает совмещение параметров θ_{enc} и θ_{hl} .

Неизменяемыми даже в режиме обучения на данном этапе реализации архитектуры $NSLP$ остаются нейросимвольная объектная модель NS и модель привязки MM . Это связано с необходимостью стабилизации процесса обучения и, в целом, с возможностью использовать

наборы данных общего назначения или собранные случайной стратегией для эффективного предобучения этих моделей.

Далее подробнее будет рассматриваться реализация режима обучения при работе архитектуры NSLP, осуществляющего одновременное обучение и планирование. В разделе 2.4 будет отдельно рассмотрен концептуальный уровень с особенностями использования языковых моделей. А различным вариантам реализации нейросимвольной объектной модели, составляющей основу уровня нейросимвольной интеграции, посвящены главы 4 и 6.

2.3.3 Планирование и обучение в нейросимвольной архитектуре

Как упоминалось в разделе 1.2.3, во многих системах агент-среда полное состояние s_t , однозначно характеризующее поведение системы и обладающее марковскими свойствами, недоступно агенту. В этих случаях вводится дополнительное промежуточное представление данных, которое называется *наблюдением* агента o_t , играющее роль сенсорной системы, обрабатывающей первичный поток данных. Для этого определяется функция наблюдений $\Omega : S \times A \rightarrow \Pi(O)$, задающая распределение вероятностей для наблюдений в текущем состоянии.

В общем виде для учета наблюдений в определении стратегии и полезности вводится так называемый частично наблюдаемый марковский процесс принятия решений [343], что является естественным в случае рассмотрения безмодельных методов обучения. В случае алгоритмов, основанных на модели, которая обычно строится для функции переходов истинных состояний, возникает необходимость вводить отдельную подзадачу восстановления состояния по наблюдению агента с помощью кодировщика *enc* (см. схему на рисунке 2.9). Формально обычно для этого используется байесовский подход для задания условной зависимости распределения над состояниями от распределения над наблюдениями и вариант стохастической динамики, когда предсказываемое наблюдение выбирается по некоторому распределению $o_t \sim p(o_t|s_t)$.

На рисунке 2.10 представлено несколько типичных вариантов построения таких байесовских графических моделей для описания модели мира M_O или латентной модели динамики M_L [386].

В случае частично наблюдаемой среды стратегию можно определить и напрямую на множестве наблюдений $\pi(o|s)$. 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 t времени и генерирует новое действие a_t в предположении, что вся существенная информация для принятия решения содержится в состоянии s_t , которое он определяет на основе текущего наблюдения o_t . Вероятность пребывания в следующем состоянии среды $b_{t+1}(s_{t+1})$ (belief state, предполагаемое состояние) определяется агентом по текущему наблюдению o_t в

соответствии с байесовским правилом для условных вероятностей:

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

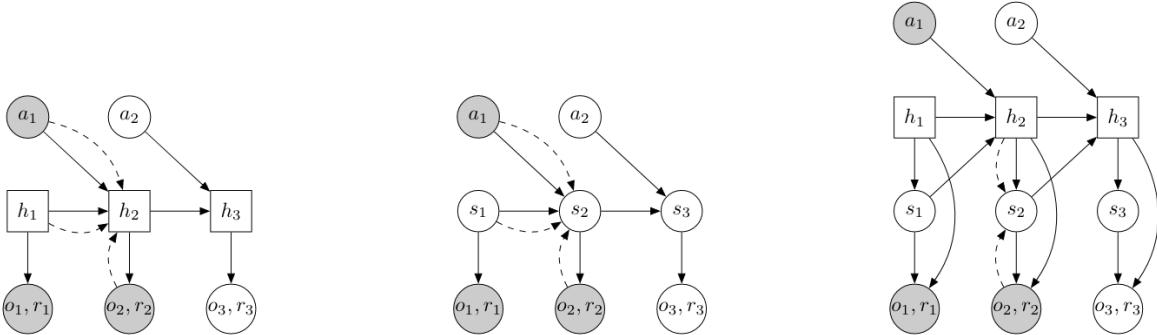


Рисунок 2.10 — Байесовские графические модели латентной динамики внешней среды. Первые два наблюдения (серые) подаются на модель, и она предсказывает третье наблюдение. Сплошные линии — процесс генерации, пунктирные — вывод модели. Слева представлена детерминированная модель с использованием рекуррентной нейронной сети, где h_t — скрытый вектор модели, моделирующий информационное состояние среды. Посередине представлен вариант со стохастическими состояниями s_t (SSM, space-state model, модель в пространстве состояний), где переходы полностью стохастичны. Справа — гибридный вариант с детерминированной и стохастической частью (RSSM, recurrent space-state model).

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 :

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

Однако в дальнейшем при рассмотрении задачи обучения с подкреплением на основе модели естественно предположить, что агент в модель мира включает обратную функцию наблюдений $enc = \Omega^{-1}$, по которой возможно восстановить предполагаемое текущее состояние среды, т.е. включает в модель оценку вероятности $b(s|o)$. Имея в стохастическом случае оценку текущего состояния s , возможно использовать исходную постановку полностью наблюдаемого марковского процесса принятия решений с заменой истинного состояния его оценкой с помощью функции enc .

Иерархическая постановка и параметризация

Далее определим иерархию на множестве действий в соответствии с концепцией полумарковского процесса принятия решений и подхода умений [143].

Определение 2.3.1 (определение умения). *Дляющееся во времени действие (умение) κ – это тройка $< I_\kappa, \pi_\kappa, \beta_\kappa >$, где $I_\kappa \subseteq O$ – инициирующее множество наблюдений, π_κ – стратегия, реализующая данное умение, $\beta_\kappa : O \rightarrow \{0,1\}$ – терминальная функция, завершающая реализацию умения.*

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 стратегию $\pi(\theta_{hl}) = \pi_\kappa$ на множестве умений (высокоуровневая стратегия), используя предобученные заранее (или обновляемые в итеративном режиме) внутренние стратегии для каждого умения π_κ (низкоуровневые стратегии) и функции завершения для каждого умения β_κ . Не снижая общности постановки задачи, можно считать, что инициирующие множества для всех умений включают все возможные наблюдения $I_\kappa = O$. Стратегия π_κ и функцию завершения β_κ параметризуются с помощью наборов параметров θ (что в данном параграфе будет соответствовать параметрам θ_{hl} из описания архитектуры NSLP) и ϑ соответственно. В иерархической постановке цель агента – максимизировать отдачу, начиная с предполагаемого состояния b_0 и умения κ_0 , – записывается в следующем виде:

$$\mathbb{E}_{\kappa, \theta} \left[\sum_t \gamma^t \sum_s b_t(s) R(s, a_t) \middle| b_0, \kappa_0 \right] \rightarrow \max_{\theta, \vartheta}$$

Пусть полезность выполнения конкретного действия a в рамках умения κ в предполагаемом состоянии $b(s)$ – это Q_a -функция, а полезность самого умения κ в $b(s)$ – это 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 штрих будет обозначать следующий момент времени):

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

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

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

Объектная 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 конкретному целевому предмету. Формально такая декомпозиция для марковского процесса принятия решений описывается в так называемой объектно-ориентированной постановке [184; 258]. В задаче одновременного обучения и планирования будет рассматриваться случай, когда взаимосвязи объектов не существенны для принятия решений агентом и принципиально только наличие тех или иных предметов в сцене. То есть в данном параграфе будет обсуждаться вопрос использование информации о сущностях $\{e_1, \dots, e_n\}$, а не всего графа сцены G_t (см. рисунок 2.9).

Итак, пусть и состояние среды s , и наблюдение агента o представляют собой некоторое множество независимых объектов, которые относятся к конечному числу классов $C = \{c_1, c_2, \dots, c_k\}$. Каждый класс характеризуется своим набором атрибутов или признаков $\{f_1^c, f_2^c, \dots, f_{n_c}^c\}$, а объект $e \in E$ класса $c(e) \in C$ описывается конкретными значениями данных признаков $e = \{d_1^c, d_2^c, \dots, d_{n_c}^c\}$. 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 $NS : O \rightarrow 2^E$, которая в данном параграфе будет считаться заранее заданной (см. вопросы ее реализации в главе 4).

В предположении независимого присутствия предметов в среде в текущем наблюдении возможна декомпозиция функции переходов и функции вознаграждений по отдельным классам объектов: $T = \{T_{c_i} | c_i \in C\}$, $R = \{R_{c_i} | c_i \in C\}$. 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 соотношения на полезность не всего состояния или наблюдения, а на полезность конкретного объекта, имеющегося в текущем наблюдении. Все соотношения на функции полезности, определенные в предыдущем разделе, остаются в силе с той лишь поправкой, что в текущем наблюдении агент выбирает действие «жадно», в соответствии с наибольшей полезностью конкретного объекта $Q_a(b, \kappa, a) = \arg \max_{e \in o} Q_a(e, \kappa, a_{c(e)})$. Предполагается, что умения агента не поддаются аналогичной декомпозиции и зависят от всего наблюдения целиком. Более подробно вопросы декомпозиции функции полезности рассмотрены в разделе 4.3.

Обновление плана поведения

В задаче одновременного обучения и планирования агент автоматически строит обновляемую латентную модель среды $M_L = \langle \hat{T}, \hat{R}, \hat{\Omega} \rangle$. Рассмотрим случай, когда модели M_L и M_O совпадают, то есть модель M_L будет иметь доступ не к состоянию s_t , а к уже декомпозированному представлению после модели NS . В этом случае латентный план B^π будет совпадать с объектным планом P , а стратегия π будет реализовываться планировщиком *planner*. Таким образом, модель M_L позволяет находить план B^π достижения цели, задаваемой в данной среде с помощью формулировки задачи $task = \langle R, I \rangle$ (см. предыдущий параграф), за счет моделирования переходов при некоторой модельной стратегии π :

$$B^\pi = \langle o_0, r_0, a_0, o_1, r_1, a_1, \dots, a_{l-1}, o_l \rangle,$$

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 \hat{R} подразумеваются приближенные (аппроксимируемые) значения функций переходов, наблюдений и вознаграждений, соответственно. План поведения агента, таким образом, составляется «жадным» алгоритмом в точности до небольшой поправки, отвечающей за исследование среды, в предположении корректности текущего приближения модели $M_L = \langle \hat{T}, \hat{R}, \hat{\Omega} \rangle$. Далее будет предполагаться, что $\hat{\Omega} = enc$ и модель будет состоять только из двух первых компонент кортежа.

Под процессом обучения агента будет пониматься итерационное обновление модели $M_L = \langle \hat{T}, \hat{R} \rangle$, 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 (см. также раздел 1.3.4). Ниже представлены краткие алгоритмические схемы для этих вариантов. Как и в предыдущем параграфе весь собранный агентом опыт будет представлен в виде множества прецедентов $\mathcal{D} = \{(o_t, r_t, a_t)\}_{t=1}^T$. Траекторией называется некоторая последовательность таких прецедентов в порядке их формирования при взаимодействии со средой.

Первый вариант интеграции представляет собой обучение с использованием планируемых («воображаемых») траекторий [237]:

1. Агент предсказывает («воображает») траектории с некоторого состояния s_t , используя модель $M_L = \langle \hat{T}, \hat{R} \rangle$.
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 \mathcal{D} агент обновляет модель среды $M_L = \langle \hat{T}, \hat{R} \rangle$.
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 $M_L = \langle \hat{T}, \hat{R} \rangle$.
2. Агент использует текущую стратегию для продолжения траекторий в процессе взаимодействия со средой.
3. По собранному опыту \mathcal{D} агент обновляет модель и стратегию, одновременно выбирая критерий остановки планирования и запуска интерактивной стратегии.

Обучение с имитацией эпизодов возможно только при наличии копий среды, в которых агент имитирует свое поведение (см. подробнее раздел 3.3):

1. Используется возможность запускать симуляции действий агента в среде, чтобы с текущего шага проиграть некоторое количество эпизодов, для обновления этой модели.
2. На основе обновленной модели определяется полезность состояний и выбирается действие, выполняемое в среде.
3. Обновляется стратегия с использованием выбранного действие в качестве эталонного.

В используемой в архитектуре NSLP реализации одновременного обучения и планирования организуется иерархия для разделения применения модели и интерактивной стратегии:

1. Агент на верхнем уровне иерархии действия использует модель $M_L = \langle \hat{T}, \hat{R} \rangle$ для получения плана на множестве умений.
2. Стратегия каждого умения π_k формируется в интерактивном режиме в процессе взаимодействия со средой (возможно, на фазе предобучения).
3. Набранный опыт \mathcal{D} используется агентом для одновременного уточнения модели на умениях M_L и стратегий каждого умения π_k .

Принимая во внимание все приведенные уточнения реализации одновременного обучения и планирования, получается иерархическая постановка обучения с подкреплением на основе модели, в которой агенту необходимо максимизировать получаемую в рамках эпизода отдачу с возможностью декомпозиции наблюдения по предметному принципу, автоматическому построению стратегий умений и одновременному автоматическому формированию модели среды, используемой для планирования на множестве умений.

2.3.4 Иерархическое обучение с использованием объектной модели среды

Одновременное планирование и обучение

Более подробно рассмотрим формальные аспекты решения задачи одновременного обучения и планирования на сенсорном уровне нейросимвольной архитектуры когнитивного агента (см. рисунок 2.9). Подсистема обучения агента делится на две составляющие, как это принято в теории обучения с подкреплением (см. раздел 1.1). Критик обновляет функцию полезности действия Q_a , а актор формирует стратегию π_k в рамках текущего умения. Цикл взаимодействия NSLP агента со средой будет выглядеть следующим образом:

1. Агент использует текущую модель M_L для того, чтобы сформировать план $B^\pi = < o_0, r_0, a_0, o_1, r_1, a_1, \dots, a_{l-1}, o_l >$ на множество умений с помощью процедуры MLplan (реализуемой планировщиком *planner*). 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 предполагается, что план B^π является низкоуровневым.
2. В соответствии с планом B^π агент выбирает текущее умение κ_t .
3. Агент получает из среды текущее наблюдение, которое с помощью функции NS переводится в набор сущностей $o_t \rightarrow \{e_1, \dots, e_n\}$.
4. В соответствии со стратегией π_k 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. В режиме обучения агент выполняет оценку выполненного действия и самого умения с помощью критика, а затем обновляет параметры аппроксиматоров критика и актора при помощи процедуры MLlearn.
7. Если текущее умение завершилось, выполняется перепланирование, и затем происходит переход к шагу 3.

Ниже представлены принципы работы процедур MLplan и MLlearn. При планировании агент использует модель $M_L = < \hat{T}, \hat{R} >$ для построения плана своего поведения. В качестве примера в данном параграфе 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 том случае, когда для выполнения выбирается действие (умение), для которого в модели не известно следующее наблюдение, образуются новые узлы с соответствующими объектами, выделенными из наблюдения, полученного из среды.

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

1. Выбирается 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, $\eta \in \mathbb{R}$ — константа, N — счетчики.
2. Производится переход по дереву $e' \leftarrow \hat{T}_c(\kappa)$, $r \leftarrow \hat{R}_c(\kappa)$, где c — класс объекта e .
3. Производится планирование для следующего объекта с подсчетом получаемого вознаграждения $\tilde{R} \leftarrow r + \gamma MLplan(M,e')$.
4. Обновляются счетчики $N(e,\kappa) \leftarrow N(e,\kappa) + 1$, $N(e) \leftarrow N(e) + 1$.
5. Настраивается модель — обновляется полезность умения $Q_{\kappa}(b,\kappa) \leftarrow Q_{\kappa}(b,\kappa) + \frac{\tilde{R} - Q_{\kappa}(b,\kappa)}{N(e,\kappa)}$, где b — предполагаемое состояние, для которого выделяется объект e .

В процедуре обучения критика и актора MLlearn производится обновление как критерия достижения подцели β_{κ} , так и стратегии π_{κ} конкретного выбранного на верхнем уровне иерархии умения. 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 $a \sim \pi_{\kappa,\vartheta}(a|o)$.
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:

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

где α_{Θ} и α_{ϑ} — шаги обучения.

5. Обновляется 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 их обновление происходит во время обновления самой модели. Далее будет 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).

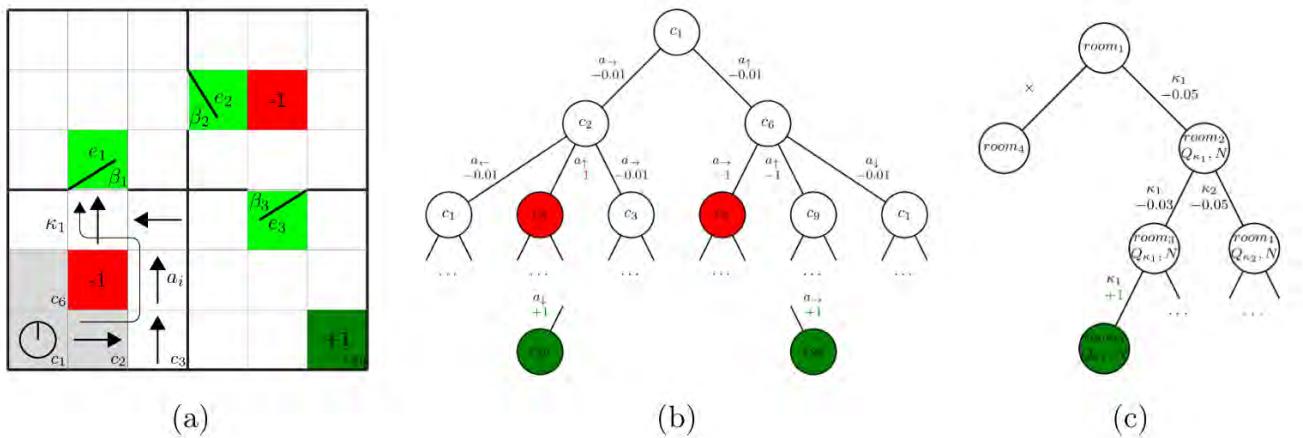


Рисунок 2.11 — (а) — модельный пример перемещения агента по клеточной среде с комнатами, открывающимися проходами между ними и опасными состояниями, в которых завершается эпизод взаимодействия. (б) — модель, соответствующая марковскому процессу принятия решения без иерархической декомпозиции. (с) — модель агента с умениями

Модельный пример

В качестве модельного примера рассматривается задача обучения навигации до некоторой заданной целевой точки в клеточной среде с препятствиями и управляемыми объектами (дверьми) (см. рисунок 2.11а). Среда организована в виде комнат с узкими проходами между ними таким образом, чтобы поддерживать формирование двухуровневой иерархии действий. Верхний уровень — 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.

В данном примере наблюдение агента o_t — это область видимости вокруг агента радиусом в одну клетку (на рисунке обозначены серые цветом). Множество действий агента A состоит из четырех действий: поворот на 90 градусов по часовой стрелке или против часовой стрелки, проход прямо, открытие двери. На верхнем уровне иерархии агенту доступны умения κ_i , состоящий из действий a_i . Подцели β_i , в которых завершаются умения являются клетки, обозначенные светло-зеленым цветом. Выделяемыми с помощью функции NS объектами e_i в среде являются сами комнаты, двери каждой комнаты, принадлежащие к 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 красные клетки -1 с завершением эпизода, за достижение цели (зеленая клетка) начисляется $+1$. В качестве выбирамой параметризации используется линейная модель, взвешивающая признаки, соответствующие объектам, выделяемым по наблюдению.

Латентная (в данном случае объектная) модель M_L , которую строит и обновляет агент, представлена на рисунке 2.11b. Для сравнения на рисунке 2.11b показана модель, которую агент аппроксимировал бы без иерархического представления действий. Процедуры планирования $MLplan(M_L, e)$ и обучения $MLlearn$ для агента в представленном примере будут выглядеть следующим образом.

Шаги процедуры $MLplan$:

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 (т.к. модель обновляется в процессе обучения, она может не соответствовать итоговой модели на рисунке 3в) 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 $Q_\kappa(b, \kappa)$.

Шаги процедуры $MLLearn$:

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 $\beta_{\kappa, \theta}$ подцель достигнута, то в соответствии с высокоуровневым планом выбирается новое умение κ .

Теоремы о градиенте стратегии и полезности в NSLP

В предыдущем параграфе была представлена общая схема взаимодействия агента со средой в режиме одновременного обучения и планирования, где для обучения агента необходимо знание целевого значения критика g и градиента функции полезности стратегии ∇J . Ниже представлены теоремы о вычислении этих значений и краткие выкладки по их доказательству. Схемы представленных доказательств аналогичны предложенным в работе [143].

Теорема 2.3.1 (о градиенте критика умений). *Для фиксированного множества марковских умений со стохастической реализациющей стратегией, дифференцируемой по параметрам θ ,*

градиент ожидаемой дисконтированной отдачи по параметрам θ с начальными условиями (b_0, κ_0) равен

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

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) .

Доказательство. В начале будет дано выражение для градиента полезности умения $Q_{\kappa}(b, \kappa)$ относительно параметров реализующей его стратегии θ :

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

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 будет обозначаться как $p(o | a, b)$. С использованием выражения в определении поправленной полезности \tilde{Q}_{κ} , легко выписывается ее градиент:

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

При подстановке этого значения в выражение для градиента полезности умения получится:

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

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 выражение, которое и требовалось доказать. \square

В предыдущем параграфе для обновления параметров генератора подцелей было указано на необходимость вычисления градиента $\nabla_{\theta} \tilde{Q}_{\kappa}$.

Теорема 2.3.2 (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

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

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) .

Доказательство. Определение для этой функции полезности дает следующее выражение:

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

Здесь также заметно наличие рекурсии и использование структуры расширенного марковского процесса принятия решений приводит к получению необходимого выражения по аналогии с теоремой 2.3.1. \square

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

2.3.5 Принцип эквивалентности и объектная декомпозиция

Как уже было неоднократно отмечено в предыдущих разделах, в настоящее время обучение с подкреплением без использования модели среды показывает впечатляющие результаты для многих задач управления поведением когнитивных агентов и их групп [4; 323; 445; 51; 74]. Однако большинство из используемых в этой области методов требуют чрезвычайно большого количества эпизодов взаимодействия агента со средой. Эта так называемая проблема эффективности выборок является одним из ключевых препятствий на пути развития новых методов и их внедрения в практику. Среди подходов к решению этой проблемы необходимо отметить имитационное обучение (с использованием заранее подготовленных демонстраций) [5; 524] и обучение на основе модели [443; 54]. Последний подход перспективен также и по той причине, что допускает возможность использования эффективных методов планирования и поиска по графу марковского процесса принятия решений [420; 567].

Построение модели с использованием аппроксиматоров является вычислительно затратной процедурой, и эффективные методы ее решения, не использующие заранее подготовленные и набранные агентом данные, только разрабатываются. Как было отмечено в предыдущем параграфе здесь перспективным является подход по использованию объектного представления состояний среды [230; 258], который позволяет декомпозировать модель среды на частные объектные модели. В этом параграфе будет уточнена объектная формулировка

задачи обучения с подкреплением на основе модели. Используются недавние результаты по свойствам эквивалентности моделей по функции полезности [616] для уточнения оптимизационной постановки задачи обучения. Основными результатами работы являются сведение оптимизационной задачи к иерархическому случаю и использование как и в предыдущем параграфе подхода умений [596], для которого существуют эффективные реализации процесса обучения [143; 356].

Как это было описано в 2.3.3 будет использоваться объектная декомпозиция. В том случае, когда всю информацию об объектах удается закодировать в признаках, в том числе информацию об их взаимодействии с другими объектами, модель среды может быть декомпозирована пообъектно: $\tilde{T} = \{\tilde{T}_{c_i} | c_i \in C\}$, $\tilde{R} = \{\tilde{R}_{c_i} | c_i \in C\}$. Такая декомпозиция естественным образом продолжается на множество действий и функцию полезности, т. е. в том случае, когда выполняется условие пообъектного разделения условий и эффектов каждого действия, множество действий разбивается на подмножества для каждого класса объектов: $A = \{A_{c_i} | c_i \in C\}$. Классовое действие $a_{c_i} \in A_{c_i}$ меняет свойства только одного объекта $e \in c_i$, $e \in s$. Аналогично можно проследить вознаграждения, получаемые для каждого класса объектов отдельно, и, таким образом, ввести объектную параметризацию функции полезности: $V^\pi(e)$, $e \in s$.

Принцип эквивалентности модели по полезности

Для формулировки полезных свойств объектного представления модели будет использоваться принцип эквивалентности модели по полезности [616]. Для начала необходимо отметить, что любая модель $M_O = < \hat{T}, \hat{R} >$ (далее просто M) индуцирует оператор Беллмана в соответствии с выражением 1.5, где функциями переходов и вознаграждения служат элементы модели M .

Определение 2.3.2 (Эквивалентность моделей). *Пусть \mathcal{P} — множество стратегий, а \mathcal{V} — множество некоторых функций (полезности). Две модели $M = < T, R >$ и $\tilde{M} = < \tilde{T}, \tilde{R} >$ называются эквивалентными по полезности относительно \mathcal{P} и \mathcal{V} тогда и только тогда, когда*

$$\mathcal{B}_\pi[V] = \tilde{\mathcal{B}}_\pi[V], \forall \pi \in \mathcal{P}, V \in \mathcal{V},$$

где \mathcal{B}_π и $\tilde{\mathcal{B}}_\pi$ — операторы Беллмана, индуцированные моделью M и \tilde{M} , соответственно.

Данное определение приводит к заданию функциональных классов моделей, в которых элементы являются неразличимыми с точки зрения структуры функции полезности, задаваемой уравнением Беллмана.

Определение 2.3.3 (Пространство эквивалентности). *Пусть \mathcal{P} — множество стратегий, \mathcal{V} — множество некоторых функций (полезности), а \mathcal{M} — пространство моделей. При заданной некоторой модели m $\mathcal{M}_m(\mathcal{P}, \mathcal{V})$ называется пространством эквивалентных*

по полезности моделей, каждая из которых эквивалентна по полезности модели m относительно \mathcal{P} и \mathcal{V} .

В обучении с подкреплением на основе модели целью агента является в том числе и поиск оптимальной модели $m^* = \langle T, R \rangle$ с истинными функциями переходов и вознаграждения. Если предполагается использование модели именно для более быстрой оценки стратегии, то агенту достаточно найти любую модель, которая эквивалентна по полезности оптимальной модели m^* , т.е. найти $m \in \mathcal{M}_{m^*}(\mathcal{P}, \mathcal{V}) \equiv \mathcal{M}(\mathcal{P}, \mathcal{V})$. Для этого оптимизационную задачу, которая ставится изначально как обычное обучение с учителем

$$\begin{cases} \arg \min_{\hat{R}} \mathbb{E}_{(s,a) \sim \mathcal{D}} [(R(s,a) - \hat{R}(s,a))^2] \\ \arg \min_{\hat{T}} \mathbb{E}_{(s,a) \sim \mathcal{D}} [D_{KL}(T(\cdot|s,a) || \hat{T}(\cdot|s,a))] \end{cases}, \quad (2.8)$$

можно переписать, объединив минимизирующие функционалы по функциям вознаграждения и переходов в один функционал для уравнения Беллмана:

$$\arg \min_{\tilde{m}} \sum_{\pi \in \mathcal{P}} \sum_{V \in \mathcal{V}} \left\| \mathcal{B}_\pi[V] - \tilde{\mathcal{B}}_\pi[V] \right\|.$$

Соответствующая функция потерь уже явно зависит от выбора стратегии и соответствующей функции полезности, что позволяет, во-первых, сократить пространство поиска, а во-вторых, использовать свойство функциональной эквивалентности по полезности моделей. Основной трудностью в данном случае является определение пространств \mathcal{P} и \mathcal{V} , по которым проходит сравнение двух операторов Беллмана. В работе [616] предлагается несколько вариантов их определения, в том числе поточечной линейной оболочки для \mathcal{P} и параметризованного пространства аппроксиматоров для \mathcal{V} .

Принцип эквивалентности и объектная декомпозиция

Далее будет рассмотрена объектная декомпозиция модели, чтобы переформулировать оптимизационную задачу и сравнить решения, получаемые при использовании объектного представления и без него. Так как ранее была введена объектная параметризацию всех функций, участвующих в определении оператора Беллмана, то его объектная версия будет выглядеть следующим образом:

$$\mathcal{B}_\pi[V](e) = E_{a(e) \sim \pi, s' \sim T} \left[R(e, a) + \gamma V(e') \mid e' \in s' \right].$$

Особенностью данного оператора является то, что выполнение действия для агента становится двухэтапным: вначале агент должен определить объект $e \in s$, с которым он будет взаимодействовать, а затем уже выбрать действие из множества доступных действий A_c , определенных для класса данного объекта $c(e)$. Таким образом, получается иерархический

вариант обучения с подкреплением, для которого обычно используется формулировка абстрактных действий в виде умений [599] (см. предыдущий параграф).

В соответствии с определением 2.3.1 пусть $\kappa(c) = \langle I_c, \pi_c, \beta_c \rangle$ — делящееся во времени действие, или умение, для которого $I_c \subseteq S$ — инициирующее множество состояний, такое, что $\forall s \in S, e \in s, c(e) = c$, π_c — стратегия, реализующая данное умение, такая, что $\forall a \sim \pi_c, a \in A_c, c(a) = c$, $\beta_c : S \rightarrow \{0,1\}$ — это терминальная функция, завершающая данное умение. Каждое умение $\kappa(c)$ задает типичную стратегию действия агента с любым объектом данного класса c . Умение может быть и одношаговым, когда $\beta_c(s_{t+1}) = 1, \forall s_t \in I_c$.

Иерархическая формулировка объектного обучения с подкреплением с использованием умений приводит к следующей реализации двухуровневой стратегии. На верхнем уровне агент на каждом шаге определяет объект определенного класса, с которым он будет взаимодействовать, — это стратегия на умениях $\pi_\kappa : S \rightarrow C$. Второй уровень представляет собой стратегию π_c , которая задает последовательность действий с объектом класса c , реализуемую умением $\kappa(c)$. В простейшем случае можно считать, что для каждого класса объектов существует только одно умение.

Для иерархического случая уравнение Беллмана для стратегии на множестве умений практически не изменится за исключением учета длительности самого умения, равной τ :

$$\mathcal{B}_{\pi_\kappa}[V](s) = E_{\kappa(c) \sim \pi_\kappa, s' \sim T} \left[R(s, \kappa(c)) + \gamma^\tau V(s') \middle| s \in I_c(\kappa) \right].$$

Определение 2.3.4 (Частичная модель). Частной моделью называется модель $t(c) = \langle T_c, R_c \rangle$, которая характеризует часть марковского процесса принятия решений задающего функции переходов и вознаграждений для конкретного класса объектов c .

Оптимизационная задача для объектно-центричного обучения с подкреплением на основе модели будет включать в себя, помимо стандартной максимизации ожидаемой отдачи, $k + 1$ подзадачи минимизации, отвечающие за поиск оператора Беллмана в рамках эквивалентных по полезности k частичных моделей и одной общей (для умений):

$$\begin{cases} \text{argmax}_{\pi_\kappa} E \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \kappa_t) \right], \\ \text{argmin}_{\tilde{m}} \sum_{\pi_\kappa \in \mathcal{P}_\kappa} \sum_{V \in \mathcal{V}_\kappa} \left\| \mathcal{B}_{\pi_\kappa}[V] - \tilde{\mathcal{B}}_{\pi_\kappa}[V] \right\|, \\ \text{argmin}_{\tilde{m}_{c_1}} \sum_{\pi_{c_1} \in \mathcal{P}_{c_1}} \sum_{V \in \mathcal{V}_{c_1}} \left\| \mathcal{B}_{\pi_{c_1}}[V] - \tilde{\mathcal{B}}_{\pi_{c_1}}[V] \right\|, \\ \dots \\ \text{argmin}_{\tilde{m}_{c_k}} \sum_{\pi_{c_k} \in \mathcal{P}_{c_k}} \sum_{V \in \mathcal{V}_{c_k}} \left\| \mathcal{B}_{\pi_{c_k}}[V] - \tilde{\mathcal{B}}_{\pi_{c_k}}[V] \right\|. \end{cases} \quad (2.9)$$

Утверждение 2.3.1. $k + 1$ подзадачи минимизации ошибки в определении оператора Беллмана могут решаться параллельно с использованием одной и той же памяти предыдентов или траекторий \mathcal{D} , для каждого состояний которой применена функция выделения объектов Φ .

Замечание 2.3.1 (о локальной организации [683]). Полученный результат справедлив только в том случае, когда существует локальность эффектов каждого действия. Это означает, что изменение свойств одних объектов не оказывается на изменении свойств

других. Это достаточно частный, но важный случай сред, пример которых будет рассмотрен ниже.

В качестве практической реализации подсчета близости модели к оптимальной эффективно использовать эмпирическую версию расстояния между предсказаниями полезности для всех состояний, которые встречаются для некоторой сгенерированной выборки:

$$\mathcal{L}(m^*, \tilde{m}) = \sum_{\pi \in \mathcal{P}} \sum_{V \in \mathcal{V}} \sum_{s \sim \mathcal{D}} \left[\frac{\sum_{\mathcal{D}, s_i=s} R(s_i) + \gamma V(s'_i)}{N_{\mathcal{D}}(s_i)} - \tilde{\mathcal{B}}_{\pi}[V](s) \right],$$

где $N_{\mathcal{D}}(s_i)$ — число вхождений состояния s_i в выборку \mathcal{D} .

Итоговый алгоритм обучения когнитивного агента с использованием объектного представления в архитектуре NSLP будет выглядеть следующим образом:

1. Инициализация верхнеуровневой стратегии агента π , объектных умений π_{c_i} , общей модели \tilde{m} и частных моделей \tilde{m}_{c_i} .
2. Генерация дополнительного опыта агента \mathcal{D} , полученного в среде с использованием стратегий π и π_{c_i} .
3. Обновление общей и частных моделей с использованием оптимизационной задачи 2.9.
4. Решение задачи планирования (определения субоптимальных функций полезности для умений V^i и общей стратегии V^k) по общей и частным моделям.
5. Использование «жадных» по полезностям V^i и V^k стратегий в качестве новой высокоуровневой стратегии π и объектных умений π_{c_i} . Переход к шагу 2.

Модельный пример

Далее будет рассмотрена известная в области исследования методов обучения с подкреплением на основе модели среды Crafter [308] (см. в том числе раздел 3.1). Это клеточная среда с богатым набором подзадач (см. рисунок 2.12), каждая из которых представляет собой конструирование некоторого объекта в среде. Наблюдение агента o представляет собой RGB-изображение размером 64×64 , часть из которого отведена под представление некоторой окрестности среды вокруг агента (вся информация сгруппирована в клетки размеров 9×7), а часть — под представление собранных агентом ресурсов («инвентарь») и его собственных свойств (насыщение, здоровье и т. д.). Агенту доступны действия по перемещению и взаимодействию с имеющимися объектами (конструирование предметов, уничтожение противников и т. д.). Данная среда хорошо подходит для объектного представления. Функция NS выделения объектов может работать поклеточно: с каждой клеткой сопоставляется один из 19 типов объектов, каждый из которых может характеризоваться плотным векторным представлением.

Частными моделями для данной среды будут являться модели изменения признаков каждого объекта, который детектируется в соответствующих клетках (за исключением типов, отвечающих за подстилающую поверхность: траву, камни и т.п.). Умениями агента будут являться те действия, которые меняют характеристики данных объектов, соответствующие им оптимальные стратегии могут задаваться заранее (например, рубка дерева или уничтожение врагов).

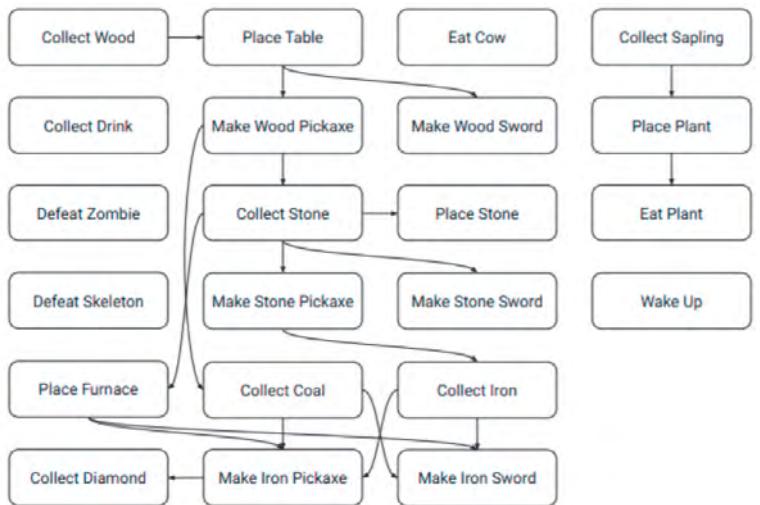


Рисунок 2.12 — Среда Crafter (слева) и список подзадач, которые выделяются в этой среде (справа).

Перспективным подходом к повышению эффективности методов обучения с подкреплением является использование модели среды. В данном разделе диссертационного исследования обсуждались особенности объектного представления состояний. С использованием понятия эквивалентности моделей по полезности была предложена иерархическая формулировка оптимизационной задачи объектно-центричного обучения с подкреплением на основе модели, сформулирован алгоритм обучения агента, а используемые понятия продемонстрированы на среде Crafter.

2.4 Планирование с языковыми моделями в NSLP

В данном разделе рассматривается концептуальный уровень принятия решений в архитектуре NSLP, где основным модулем выступает предобученная языковая модель L (см. рисунок 2.13). Большие языковые модели (БЯМ) [152; 487], которые обучаются на большом массиве не размеченных текстов, представляют собой экспертные системы общего назначения с интерфейсом запросов (prompts, «затравок») и ответов на естественном языке. Такие модели могут выполнять роль генератора гипотез H , представляющих собой описание шагов по достижению цели или выполнению некоторой инструкции. Данная гипотеза уже может служить более формальной задачей для планировщика, который генерирует символьный план P уже с использованием тех действий, которые доступны

агенту. Предполагается, что такой план уже провалидирован и, в принципе, достигает цели в начальных условиях, которые описывают текущую среду агента (графовое описание G_t).

Далее в разделе будет дан краткий обзор использования больших языковых моделей в задачах планирования (раздел 2.4.1), а затем представлена методика и метрики по оценке эффективности решения задач планирования языковыми моделями с различным числом параметров. Основные результаты, изложенные в данном разделе, представлены в публикациях [62; 66].

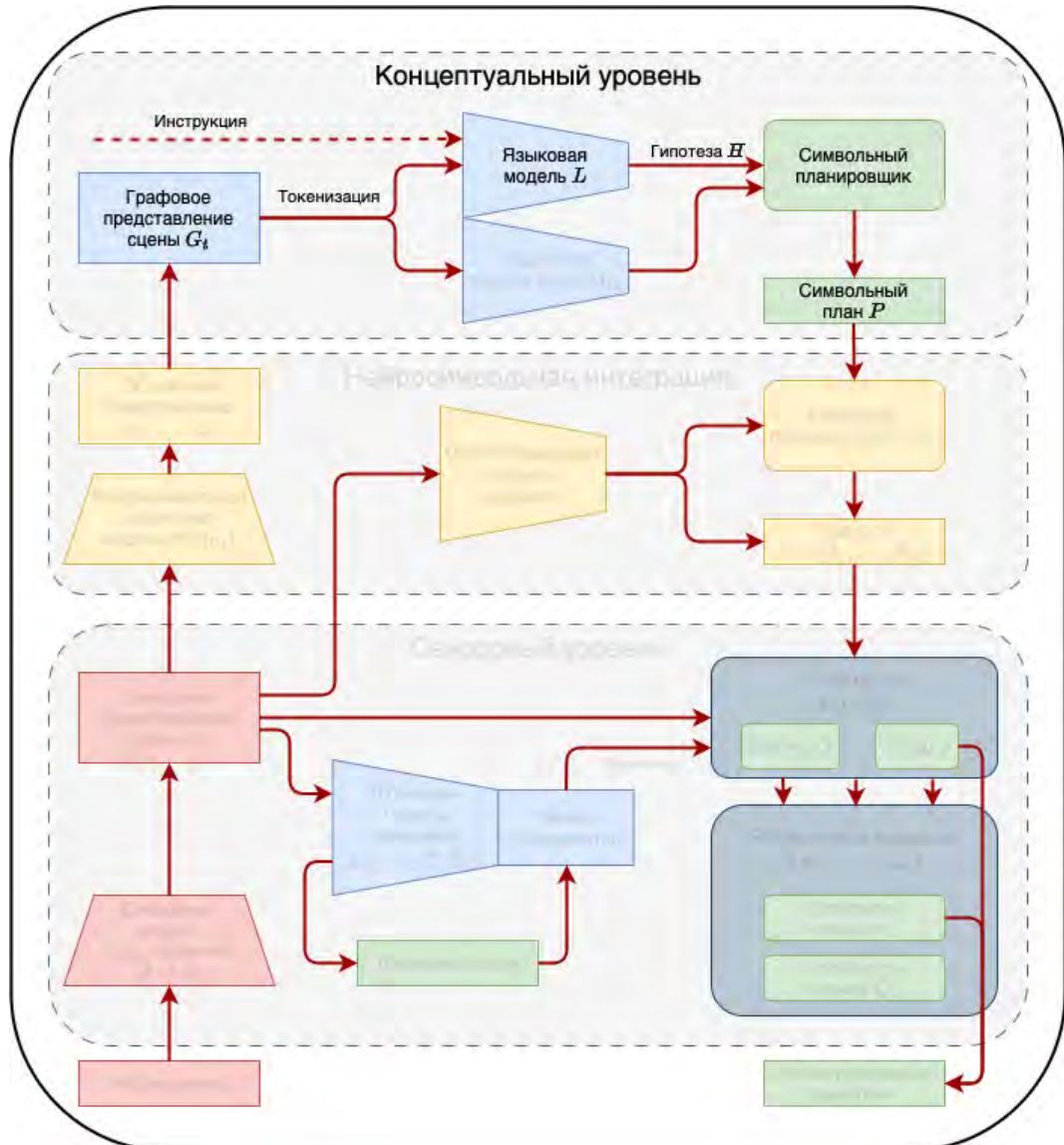


Рисунок 2.13 — Архитектура NSLP с выделенным концептуальным уровнем, на котором используются языковые модели для генерации гипотезы и составления символного плана действий.

2.4.1 Применение предобученных больших языковых моделей в задачах планирования

В последние годы возрос интерес к задачам воплощенного искусственного интеллекта (*embodied artificial intelligence*), которые, в основном, представлены либо задачами оперирования объектами в человеко-ориентированных средах (*household tasks*), либо перемещением объектов и навигацией в помещениях или на открытой местности. Отличительной чертой «воплощенных» задач является формулирование на естественном языке инструкций, описывающих выполняемую задачу или достигаемую цель и передаваемых воплощенному когнитивному агенту. Такая постановка приводит к необходимости использовать техники обработки естественного языка для перевода инструкций в вид удобный для использования воплощенного искусственного интеллекта. Существуют два подхода к решению этой задачи.

В рамках первого подхода используются специализированные модели, обученные для получения плана действия агента на основе инструкции. Примерами могут служить техника использования шаблонов возможных действий и определения аргументов этих действий [262] или модели генерации последовательности токенов (*Seq2seq*) [402].

Второй подход основан на том, что современные большие языковые модели (БЯМ) [152; 487], предобученные на больших корпусах не размеченных текстов, демонстрируют хорошие результаты в задачах, для решения которых они изначально не были спроектированы, после небольшого дообучения (*few-shot learning*) [376] или совсем без дообучения [263]. Это достигается за счет того, что такие модели хранят знания общего назначения (*common sense*). Современные работы используют это свойство БЯМ в задачах воплощенного искусственного интеллекта, например, [378].

В данном разделе подробно рассматриваются модели, относящиеся ко второму подходу к задачам воплощенного искусственного интеллекта, которые используют предобученные БЯМ для генерации плана поведения воплощенному когнитивному агенту в среде.

В работе, посвященной планировщикам без дообучения [378], предлагается использовать БЯМ для «заземления» высокоуровневой задачи, выраженной на естественном языке, на множество элементарных действий, доступных интеллектуальному агенту. В результате, по описанию задачи БЯМ должна построить план действий, приводящий к выполнению поставленной задачи, при этом подразумевается, что БЯМ не дообучается. В качестве среды, в которой действует интеллектуальный агент, используется симулятор *Virtualhome* [639]. Построение плана происходит итеративно: сначала на вход модели подается специально сформулированный запрос с описанием задачи на естественном языке, после чего модель в свободной форме генерирует описание действия, которое необходимо выполнить на первом шаге. Для полученного описания действия считается специальное векторное представление [526], для которого ищется действие из множества элементарных действий с наиболее близким векторным представлением. После этого

описание полученного действия добавляется к тексту запроса и процедура повторяется. Запрос БЯМ формулируется в виде подсказки, где в начале запроса идет пример задачи и план действий по её выполнению, а в конце добавляется описание текущей задачи. Основное внимание авторы уделяют составлению плана; при этом предполагается, что агент уже умеет выполнять элементарные действия.

В G-PlanET [477] также рассматривается использование БЯМ для генерации плана действий, однако, в отличие от планировщиков без дообучения [378], акцент делается на привязке к конкретной среде, а не только к действиям агента. Используется модификация задачи ALFRED [124], в которой сцена представляется как таблица с перечислением всех доступных на сцене объектов с их типом, положением, ориентацией и родительским объектом (на чем лежит / в чем находится данный объект). На этапе планирования таблица сцены построчно разворачивается и объединяется с описанием задачи. Полученный текст в виде запроса подается на вход БЯМ. Таким образом, в качестве запроса в G-PlanET [477] используется сгенерированное табличное представление сцены, в которой функционирует интеллектуальный агент, и описание задачи на естественном языке. План генерируется итеративно, результат текущего шага конкатенируется с запросом для этого шага и подается на вход модели для генерации следующего шага. Похожий подход используется в EA-APG (environmentally-aware action plan generation) [293], однако в нем описание сцены состоит только из перечисления объектов.

В архитектуре SayCan [233] процесс отдельного обучения агента выполнению элементарных действий является одним из ключевых этапов. Действие состоит из трех частей: стратегии, т.е. последовательности команд по перемещению интеллектуального агента или его подвижных частей; описания на естественном языке; и функции соответствия цели (affordance function), возвращающей вероятность успешного выполнения действия в текущем состоянии среды. Процесс генерации плана похож на процесс, реализованный в планировщике без дообучения [378], с тем различием, что БЯМ не генерирует описания следующего действия, а дает оценку вероятности того, что элементарное действие полезно для выполнения поставленной задачи. Такая оценка выдается для всех возможных элементарных действий и умножается на вероятность успешного выполнения действия, полученную по функции соответствия (полезности). В качестве следующего выбирается действие с максимальным значением оценки. Отличительной особенностью этой работы является то, что эксперименты проводились как в виртуальной среде, так и на робототехнических платформах в реальной среде. Также стоит отметить, что запрос состоит не из одного примера, как в планировщиках без дообучения [378], а содержит несколько примеров. Другое отличие в формировании запроса в том, что он формируется не как описание задачи с планом действий, а как диалог между пользователем и интеллектуальным агентом, в котором пользователь задает вопрос, например, «Как бы ты мог принести мне перекус?», а агент перечисляет действия, необходимые для решения поставленной задачи. Также авторы применили подход к формированию запроса, предложенный в [173]. Он предполагает добавление к запросу описания процесса решения задачи помимо самих задачи и решения. Использование такого запроса в SayCan [233] позволяет улучшить работу

модели для задач, где используется отрицание или процесс рассуждения. Ограничение такого подхода заключается в том, что, как показано в [173], улучшения демонстрируются только для БЯМ с более чем 100 млрд параметров.

В архитектуре ProgPromt [511] основная идея заключается в представлении запроса в виде Python-подобного кода. Запрос состоит из описания доступных действий в виде импортирования соответствующих программных модулей, списком с перечислением доступных объектов на сцене, примерами задачи и их выполнения в виде программных модулей. Использование такого вида запроса обосновывается тем, что БЯМ, такие как GPT-3 с 175 млрд параметров [376], обучаются в том числе и на большом количестве данных из открытых репозиториев программного кода. Похожий подход используется в модели CaP [185], которая генерирует стратегию агента. Его отличительная черта заключается в том, что получаемые программы являются полноценным исполняемым программным кодом и позволяют организовать иерархичность выполнения самих программ.

Некоторые работы, например, модель по Сократу [578] и модель внутреннего диалога [328], предлагают не конкретную модель, а подход к построению и объединению множества моделей. Так, в модели по Сократу [578] предлагается использовать предобученные модели, использующие разные модальности (звук, текст, изображение и др.), и объединять такие модели в системы, которые способны решать задачи, выходящие за рамки задач каждой отдельной модели. Это достигается за счет создания интерфейса обмена данными между моделями. В качестве примера робототехнической задачи предлагается планирование перемещения объектов на столе. Используется симулятор Pybullet⁷ для детектирования объектов на сцене. Для составления описания по изображению применяется подход ViLD [480], после чего описание сцены в виде запроса передается в БЯМ для генерации плана по аналогии с планировщиками без дообучения и SayCan. Далее план выполняется CLIPort-подобной стратегией [565; 623]. Запрос состоит из описания сцены в формате перечисления объектов python-подобным списком, примеров формулировки задач на естественном языке и их выполнения в виде псевдокода.

В архитектуре модели внутреннего диалога [328] предлагается использовать обратную связи в виде текста, полученную либо от среды (описание сцены, успешность выполнения действия), либо от пользователя (уточнение необходимого действия агента). При этом, сохраняя общий подход, в зависимости от задачи для реализации используются разные модели. Основная идея заключается в итеративном добавлении обратной связи от среды в виде текста во входной запрос, используемый для планирования БЯМ. При манипулировании с объектами (в среде с виртуальным или реальным столом) используется запрос с описанием сцены и примерами задач. Для выполнения задачи на реальной кухне запрос форматируется в виде диалога пользователя и воплощенного агента.

В работе LM-Nav [412] используется подход, подпадающий под определение модели по Сократу [578], когда для задачи навигации по тексту и изображению используются отдельно предобученные модели, соединенные в общую систему. БЯМ GPT-3 [376] используется для генерации последовательности текстовых ориентиров на основе инструкций на естественном

⁷<http://pybullet.org>

языке. Визуально-языковая модель сопоставляет текстовые ориентиры с изображениями, получаемыми агентом [400], а модель навигации по изображению [638] строит и выполняет план перемещения агента. В качестве запроса для БЯМ используются три примера извлечения текстовых ориентиров.

Классификация БЯМ для задачи планирования

Классификация рассмотренных подходов по типам используемых запросов, средам тестирования и применяемым БЯМ представлена в табл. 1. Обобщая данные, приведенные в табл. 1, необходимо заметить, что в общем виде запрос для языковой модели может состоять из следующих частей:

- Описания сцены, которое или заключается в простом перечислении доступных объектов, или дополняется приведением свойств этих объектов;
- Перечисления доступных для воплощенного агента действий;
- Примеров задач, поставленных перед воплощенным агентом;
- Примеров выполнения поставленных задач.

При этом сам запрос может выражаться или простым текстом, или в виде диалога. Помимо запроса на естественном языке, возможно использовать запросы, представляющие собой или псевдокод или выполняемый программный код, например, на языке Python.

Необходимо отметить, что задача подбора правильного запроса является непростой, и на результат могут влиять такие, казалось бы, незначительные изменения как нумерация списка действий плана и добавление символов переноса строки (см. примеры в SayCan [233]).

Все рассмотренные подходы могут быть описаны общей архитектурой использования БЯМ для задачи планирования действий воплощенного агента, представленной на Рисунке 2.14.

На первом этапе на основе инструкции на естественном языке, описывающей задачу для воплощенного агента, формируется запрос для БЯМ. В простейшем случае запрос состоит из примеров задач с планами выполнения в терминах, доступных для воплощенного агента инструкций [233; 378; 412]. Более сложная структура запроса может включать дополнительную информацию о среде, например, перечисление присутствующих объектов и их свойств [185; 293; 328; 477; 511; 578] или доступных действий агента [185; 511]. Далее запрос поступает в БЯМ, которая итеративно генерирует план поведения. Стоит заметить, что, в основном, запрос формулируется на естественном языке [233; 293; 328; 378; 412; 477; 578], однако существуют подходы, использующие для этого псевдокод [185; 578] или программный код [511].

На втором этапе агент выполняет полученный план. Такая постановка подразумевает, что план поведения генерируется полностью до начала выполнения в среде и не модифицируется в процессе выполнения. Это может привести к ситуации, когда агент застревает на одном из этапов выполнения плана, что, в свою очередь, может обернуться

невыполнением исходной задачи. Решением этой проблемы может быть использование обратной связи от среды (пунктирная стрелка на рисунке 2.14) на каждой итерации генерирования следующего действия агента.

Таблица 1 – Сравнения алгоритмов для задач воплощенного искусственного интеллекта на основе предобученных БЯМ.

Алгоритм	Языковая модель	Среда	Робот. реал.	Тип запроса	Нав.	Взаим. сред.
Zero-Shot Planners [378]	GPT-3 175B [376], Codex 12B [254]	VirtualHome [639]	-	Примеры задач и их выполнения	+	+
G-PlanET [477]	TaPEX [604]	ALFRED [124] (модификация)	-	Описание сцены, описание задачи	-	-
EA-APG [293]	GPT-3 [376]	VirtualHome [639]	-	Описание сцены, примеры задач и их выполнения	+	+
SayCan [233]	PALM 540B [487]	Естественная среда (кухня)	Everyday Robots	Примеры задач, их решения, сформулированные в виде диалога	+	+
ProgPromt [511]	GPT-3 175B [376]	VirtualHome [639]	-	Python-подобный код с описанием доступных действий, сцены, примерами задач и их выполнением	+	+
Socratic [578]	GPT-3 175B [376]	Pybullet	-	Описание сцены, примеры задач и их выполнения на псевдокоде	-	+
Inner Monologue [328]	Instruct GPT [622]	Pybullet	-	Описание сцены, примеры задач и их выполнения	-	+
Inner Monologue [328]	Instruct GPT [622]	Естественная среда (стол)	Everyday Robots	Описание сцены, примеры задач и их выполнения	-	+
Inner Monologue [328]	PALM 540B [487]	Естественная среда (кухня)	Everyday Robots	Примеры задач и их выполнения в виде диалога	+	+
CaP [185]	Codex [254], code davinci-002	Естественная среда (рисование на белой доске)	UR5e	Python-подобный код с описанием доступных действий, сцены, примерами задач и их выполнения	+	-
CaP [185]	Codex [254], code davinci-002	Естественная среда (стол)	UR5e	Python-подобный код с описанием доступных действий, сцены, примерами задач и их выполнения	+	-
CaP [185]	Codex [254], code davinci-002	Естественная среда (кухня)	Everyday Robots	Python-подобный код с описанием доступных действий, сцены, примерами задач и их выполнения	+	+
LM-Nav [412]	GPT-3 [376]	Естественная среда (улица)	Clearpath Jackal UGV	Примеры задач и их выполнения	+	-

В качестве обратной связи может использоваться информация о возможности выполнения конкретного действия [233] или отчет о корректном выполнении действия или об изменении состояния объекта [328].



Рисунок 2.14 — Общая архитектура использования больших языковых моделей (БЯМ) для задачи построения плана поведения воплощенного агента.

Рассмотренные подходы использования БЯМ для планирования поведения демонстрируют неплохие результаты как в виртуальных средах, так и при имплементации на робототехнических платформах. Тем не менее, количественное сравнение этих работ осложнено тем, что в них используются (за исключением нескольких исследований) различные среды. При этом существуют и хорошо известны задачи с установленными метриками качества и таблицами сравнений, на которых тестируются воплощенные интеллектуальные агенты, например, ALFRED [124] и TEACH [606], для следования инструкциям на естественном языке, RoomR [641] и Benchbot [151] для перестановки объектов и др. К сожалению, пока в этих бенчмарках большинство указанных работ не представлены, что во многом объясняется сложностью организации запросов в разнообразных средах с большим количеством объектов и действий.

2.4.2 Оценка предобученных БЯМ в задачах планирования

Измерение качества сгенерированных планов является сложной задачей в воплощенном искусственном интеллекте из-за мультимодального характера проблемы и неоднозначности естественного языка. Для оценки качества необходимо не только измерять степень понимания агентом естественного языка, но и способность связывать этапы плана с допустимыми действиями и с визуальной информацией из окружающей среды. Принято

оценивать планы с точки зрения их выполнимости и корректности [293; 378; 511]. Выполнимость подразумевает проверку того, что каждый шаг плана синтаксически корректен, т.е. содержит допустимые действия и объекты и может быть выполнен агентом в среде. Корректность — это оценка того, приводит ли выполнение шагов плана к достижению целевого состояния среды. Как отмечено в [378], трудно судить о корректности на основе одного измерения из так называемого золотого стандарта, поскольку одна задача может иметь несколько правильных решений (последовательности действий), поэтому используются различные сложные показатели.

Метрика максимальной по длине общей подпоследовательности (LCS) [293; 378] оценивает долю пересечения сгенерированного плана с основным истинным планом (ground true, GT). В тех случаях, когда план выполняется непосредственно в среде, используются показатели успешности и выполнения целевых условий [124; 511], которые оценивают, соответственно, полное или частичное достижение всех целевых условий, относящихся к задаче. В [477] авторы предлагают оценивать планы шаг за шагом, используя метрики на основе графиков сцен CIDEr [635] и SPICE [587], и используют метрику KeyActionScore (KAS) для извлечения ключевых действий (например, в наборе данных ALFRED [124]), необходимых для завершения задачи. Граф конечного состояния сцены, где узлы представляют объекты, а ребра — отношения между ними, используется для расчета корректности, например, в [293].

На сегодняшний день не существует единой общепринятой метрики для задач оценки качества планирования языковыми моделями. Эксперты также могут быть вовлечены в процесс оценки, как, например, это происходит в [233] и в [378]. В данном разделе диссертационного исследования предлагается использовать три показателя, которые оценивают сходство сгенерированного плана и плана GT с точки зрения различных критериев, а также проводится экспертная оценка результатов планирования для получения подробного анализа эффективности исследуемых моделей.

Опишем формально задачу генерации плана с помощью языковых моделей в соответствии с классическими условиями планирования (см. раздел 1.2). Учитывая текстовое описание цели (задачи, инструкции) τ , БЯМ должна построить символьный план достижения цели P , который представляет собой последовательность подзадач $P = (p_1, \dots, p_n)$. Подзадача p_t представляет собой пару (a_t, e_t) , $a_t \in A$, $e_t \in E$, где a_t — это действие, доступное для выполнения агентом и включающее взаимодействие с целевым объектом e_t .

Чтобы построить план с учетом оценки выполнимости подзадач, предлагается использовать БЯМ в режиме модели оценки, предложенной в подходе SayCan [233], когда модель выбирает возможное завершение текстового описания плана из ограниченного набора опций, а не генерирует текстовое описание напрямую. План достижения цели строится итеративно: на каждом шаге БЯМ выдает распределение по набору языковых описаний подзадач L . Отображение φ сопоставляет каждую возможную подзадачу p_t с соответствующим текстовым описанием $\varphi : A \times E \rightarrow L$. Здесь предполагается, что модель привязки в архитектуре NSLP задана, и для каждого действия $a \in A$, доступного для

выполнения в среде агентом, существует шаблон с его описанием на естественном языке, в который вставляется объект $e \in E$ (например, подзадача " $(\text{PickupObject}, \{obj\})$ " переходит к описанию " $\text{PickupObject } \{obj\}$ "). БЯМ оценивает все подзадачи на естественном языке $l \in L$, вычисляя вероятности их генерации в соответствии со своей моделью и выбирая оптимальную подзадачу:

$$l_t^* = \operatorname{argmax}_{l_t \in L} \mathbb{P}(l_t | f(\tau, l_1, \dots, l_{t-1})), t = \overline{1, n}, \quad (2.10)$$

где $f(x)$ — это функция запроса, которая сопоставляет входные данные x с шаблоном, специфичным для конкретной задачи. Подзадачи, полученные на предыдущих шагах алгоритма, последовательно присоединяются к входным данным функции `prompt`, которая добавляет их в строку `prompt` в соответствии с шаблоном. Результирующая последовательность языковых подзадач (l_1, \dots, l_n) может быть далее легко преобразована в исполняемый план, используя обратное отображение φ^{-1} : $s_t = \varphi^{-1}(l_t)$.

Такой подход обеспечивает выполнимость каждой подзадачи в плане, поскольку мы ограничиваем выходные данные модели определенными токенами, соответствующими подзадачам, существующими в среде. Однако это требует многократного прохождения запроса через модель для каждой подзадачи на каждом шаге алгоритма, что увеличивает время генерации плана и требует больших затрат ресурсов. Вычислительная сложность возрастает с увеличением количества доступных действий и объектов в среде. Чтобы ускорить процедуру генерации плана, было предложено дополнительно уменьшить пространство подзадач, используя набор правил, налагаемых экологическими ограничениями среды и соображениями здравого смысла.

На каждом шаге генерации плана набор подзадач L фильтруется функцией сокращения: $L' = \text{reduce}(L)$. В частности, предлагается использовать модель для оценки только тех подзадач, которые удовлетворяют правилам, разработанным для среды ALFRED [124]. Например, такие правила учитывают то, что у агента есть только один манипулятор и он не может забрать два объекта одновременно, поэтому, если ранее выбранная подзадача имеет действие «*pick up*», другие подзадачи с таким же действием отбрасываются. Кроме того, подзадачи не должны содержать объектов, которые агент в сцене не наблюдает (которые не входят в граф сцены G_t архитектуры NSLP).

Было обнаружено, что исключение некоторых избыточных подзадач из процесса оценки может значительно ускорить время вывода БЯМ, а также немногого повысить его производительность. В частности, в рассматриваемой постановке задачи имеется 159 уникальных подзадач. На каждом шаге генерации плана 159 запросов длиной ~ 1500 токенов должны пройти через модель для оценки каждой подзадачи, в то время как применение правил сокращает количество запросов в среднем до 50. Таким образом, вместо ~ 250000 токенов модель получает ~ 75000 токенов.

Используемая в качестве БЯМ авторегрессионная языковая модель обучается с функцией потерь максимального правдоподобия для моделирования вероятности последовательности токенов y , обусловленной некоторой входной последовательностью x , т.е. веса модели $\theta = \operatorname{argmax}_\theta P(y|x; \theta)$. Обученная модель затем используется в

режиме прогнозирования $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y} \in \mathbb{S}} P(\mathbf{y}|\mathbf{x}; \theta)$, где \mathbb{S} — это набор всех текстовых последовательностей.

БЯМ обучаются на больших наборах текстов и могут использоваться в режиме контекстного обучения (in-context learning), задействуя только контекстную информацию, предоставляемую в запросе. Этот подход можно использовать для запроса к БЯМ с целью генерации планов действий для выполнения задач, добавляя некоторый контекст, относящийся к данной задаче, к входной последовательности \mathbf{x} (prompt, запросу). Для цели генерации плана действий запрос обычно состоит из одной или нескольких частей: высокоуровневая постановка задачи, описание текущего состояния среды (по графу сцена G_t) и примеры аналогичных задач, которые уже были решены.

Чтобы сгенерировать полный план, языковая модель генерирует текстовую последовательность \mathbf{y} на основе входной последовательности $\mathbf{x} = f_{prompt}(Y)$. При создании полного плана с использованием языковой модели сложной задачей может оказаться сопоставление предлагаемых команд с теми, которые доступны агенту в текущей среде. Примеры возникающих в среде проблемы: 1) действие или его аргументы представлены синонимом или альтернативной фразой (например, «pick up the cup» вместо «pick up the mug»); 2) действие или его аргументы лексически некорректны (например, «cut the knife and heat it up»); 3) план генерируется в свободной форме, а не с использованием структурированного шаблона (например, «the robot should pick up the spoon and put it in the sink» вместо «pick up the spoon, then put it in the sink»); и 4) последовательность действий логически не связана (например, чтобы достать продукт из холодильника, сначала необходимо открыть холодильник). Качество генерации плана улучшается с увеличением количества примеров в запросе.

Решение проблемы с привязкой генерируемой языковой моделью гипотезы к условиям среды предлагается путем использования пошаговой генерации. В случае итеративной пошаговой генерации плана подзадачи, доступные в среде, сравниваются с подзадачей, генерируемой на каждом шаге. Сгенерированная подзадача p_g сравнивается со всеми возможными подзадачами в среде $\{p'_1, p'_2, \dots, p'_k\}$, и выбирается наиболее близкая \hat{p}' по метрике C :

$$\hat{p}' = \arg \max_{p'_i} [C(f_{emb}(p_g), f_{emb}(p'_i))], \quad (2.11)$$

где f_{emb} — функция генерации векторных представлений, а C — например, функция косинусного расстояния. В данной работе использовался подход Sentence-BERT[526] для получения векторных представлений подзадач.

Сгенерированный шаг затем добавляется в запрос, и процесс генерации продолжается итеративно до тех пор, пока не будет сгенерирована последовательность токенов, сигнализирующая об остановке процесса генерации. Преимуществом такого подхода является повышенная выполнимость сгенерированного плана, поскольку все действия с объектами могут быть привязаны к окружающей среде. Недостатком такого подхода является увеличение вычислительной сложности, которая растет экспоненциально с увеличением количества доступных действий и объектов.

Метрика качества сгенерированных планов

Чтобы измерить правильность формирования плана, было использовано несколько показателей, которые оценивают как полное совпадение планов, так и их частичное сходство. **Точность (ACC)**, т.е. точное совпадение последовательностей подзадач GT с теми, что были спрогнозированы моделью, является наиболее строгим критерием, который требует соответствия как действий, так и объектов взаимодействия для каждой подзадачи в планах:

$$ACC(S, S') = \begin{cases} 0, & \exists s_k \in S, s'_k \in S' : a_k \neq a'_k \vee o_k \neq o'_k, \\ 1, & \forall s_k \in S, s'_k \in S' : a_k = a'_k \wedge o_k = o'_k. \end{cases} \quad (2.12)$$

Была разработана метрика **Точное соответствие действий (AEM)**, чтобы оценить способность метода корректно обрабатывать особенности задачи, в частности, классифицировать ее тип. AEM требует сопоставления действий, но позволяет модели заменять объекты:

$$AEM(S, S') = \begin{cases} 0, & \exists s_k \in S, s'_k \in S' : a_k \neq a'_k, \\ 1, & \forall s_k \in S, s'_k \in S' : a_k = a'_k. \end{cases} \quad (2.13)$$

Также предлагается вычислять метрику **LCS** между двумя планами, нормализованную по максимальной длине из двух, следуя работе [378]. Однако бывают случаи, когда планы GT не позволяют определить правильность плана. Для одной задачи может быть несколько правильных планов. Например, для выполнения задачи «*put a cup with a fork in it on the counter*» агент может выполнять действия в другом порядке: поставьте чашку на прилавок, затем положите вилку в чашку или наоборот. Согласно ACC, правильным является только один фиксированный порядок подзадач. Кроме того, задачи часто могут быть сформулированы неопределенно, поэтому без представления информации из окружающей среды можно только делать предположения об их назначении и задействованных объектах. Например, явно неопределенным является описание задачи «*to acquire an odd item as place it where it is not useful*»). Таким образом, необходимо привлекать экспертов (**Human assessment (H)**) в процесс оценки плана. В представленных далее экспериментах трех человек попросили оценить, сколько правильных планов, по их мнению, сгенерировала модель, основываясь только на требованиях текстовой инструкции. Планы GT также были доступны для ознакомления. Конечное значение данного показателя получается путем деления количества правильных планов на общее количество задач и усреднения по количеству оценщиков.

Для определения подходящих языковых моделей для генерации плана было проведено исследование общедоступных предобученных моделей с большим количеством параметров. В задаче планирования поведения агента часто используется модель GPT-3 [376] с количеством параметров 175B. Однако OpenAI API для GPT-3 позволяет генерировать/оценивать только ограниченное количество токенов, что особенно критично для ресурсоемкого режима оценки

подзадач. Поэтому были рассмотрены модели без ограничений по количеству входных и выходных токенов, а также достаточно быстро работающие с ограниченным количеством требуемых вычислительных ресурсов. В качестве таких БЯМ были использованы и сравнивались три модели: GPT-J-6B (GPT-J)⁸, GPT-NeoX-20B (GPT-NeoX) [292] и OPT-30B (OPT) [482]. GPT-J — это авторегрессионная языковая модель, подобная GPT-3, предварительно обученная на наборе данных Pile [612], содержащем 825 ГБ корпуса текстов на английском языке. GPT-J содержит 6B обучаемых параметров. Архитектура модели GPT-NeoX почти повторяет модель GPT-J и была обучена на том же наборе данных, но имеет большее количество параметров (20B). OPT также является декодерной моделью из семейства GPT-3 и имеет наибольшее количество параметров (30B) в проведенных экспериментах. OPT был обучен на нескольких подмножествах Pile и некоторых других наборах данных (RoBERTa [537], CCNews [172] и др.), содержащих тексты на английском языке и небольшое количество неанглоязычных данных. Модели GPT-J и OPT используют токенизатор BPE [165], как и GPT-3, в то время как GPT-NeoX использует токенизатор, специально разработанный для этой модели.

Оценка качества моделей в задаче ALFRED

Набор данных ALFRED [124] — это эталон для оценки способности систем искусственного интеллекта понимать инструкции на человеческом языке и действовать в соответствии с ними в интерактивных визуальных средах, таких как естественная домашняя обстановка. Набор данных включает в себя демонстрацию заданий экспертами, сопровождающую инструкциями на естественном языке, в 120 различных сценах внутри помещений в среде AI2-THOR 2.0 [121]. Эти демонстрации включают в себя ситуации с частичной наблюдаемостью, длительные горизонты действий, недостаточно определенный текст инструкции на естественном языке и наличие необратимых действий. ALFRED включает в себя в общей сложности 25 743 инструкции на английском языке, описывающие 8 055 демонстраций экспертов, в результате чего суммарно получается 428 322 пары изображение-действие. В виртуальной среде агент получает эгоцентрическое наблюдение (RGB камера) и должен выполнять задачи, взаимодействуя с объектами с использованием так называемой пиксельной маски, по которой проверяется наличие объекта взаимодействия. Данный тест и набор данных предназначен для облегчения разработки систем искусственного интеллекта, которые могут переводить человеческий язык в последовательности действий и взаимодействий в визуально и физически реалистичной среде моделирования.

Чтобы создать планы GT, был создан набор данных из описаний задач и соответствующих последовательностей подзадач, извлеченных из демонстраций экспертов.

⁸GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model — <https://github.com/kingoflolz/mesh-transformer-jax>.

В ALFRED есть семь высокоуровневых действий: *PickupObject*, *putObject*, *SliceObject*, *HeatObject*, *CoolObject*, *CleanObject*, *ToggleObject*, — большинство из которых необходимо разбить на подзадачи более низкого уровня (например, чтобы нагреть объект, агент должен открыть микроволновую печь, поместить в это объект и т.д.).

В ALFRED имеется семь типов заданий, различающихся по сложности: *Pick&Place*, *Look in Light*, *Heat&Place*, *Cool&Place*, *Clean&Place*, *Pick Two&Place* и *Stack&Place*. Эти задания были разделены их на три группы по возрастающей сложности и длине плана и были выбраны по одному типу из каждой группы для экспериментов. *Pick&Place* (PP) — самый простой тип, в котором нужно взять объект и поместить его в другое место. В режиме *Heat&Place* (HP) объект необходимо дополнительном разогреть, прежде чем куда-либо поместить. В некоторых задачах объект следует разрезать ножом, что увеличивает длину плана еще на одно действие. Наиболее сложным типом задачи является *Stack&Place* (SP), в котором нужно поместить объект в подвижный контейнер и разместить их в указанном месте. Объект также можно нарезать ломтиками.

При генерации запроса был использован шаблон, предложенный в [233], который представляет собой диалог между пользователем и агентом. Данный запрос был адаптирован для задачи ALFRED. Сначала был добавлен префикс к запросу, описывающий общую постановку задачи. Затем была проведена предварительная обработка данных ALFRED с удалением знаков препинания из описаний задач и переводом их в нижний регистр. Эти описания использовались в качестве пользовательского ввода в диалоговом режиме. Далее были преобразованы планы GT: каждая подзадача в плане GT сопоставлялась с подзадачей на естественном языке с помощью функции $\phi()$.

Для каждого запрашиваемого типа задачи случайным образом выбирались n примеров планов GT и объединялись с запросом, чередуясь по типу. Обязательное описание задачи τ переформулировалось как вопрос «Human: How would you $\{\tau\}$?», а последовательность подзадач l_k в плане GT форматировалась в последовательность ‘«Robot: I would: 1. $\{l_1\}$, 2. $\{l_2\}$, ..., $n. \{l_n\}$ ».

Чтобы избежать искажений модели вследствие повторяющихся или неоднозначных примеров задач, запросы были дополнительно отредактированы вручную с заменой одних примеров другими и с добавлением большего разнообразия в классы объектов.

Запрос может содержать примеры как разных типов, так и только одного. Постановка задачи с однотипным запросом упрощается, поскольку модель «учится» на планах с похожими последовательностями действий, и это может привести к повышению производительности. Такой подход возможен, если можно каким-либо образом классифицировать типы задач перед добавлением их в запрос, например, с использованием обученной языковой модели, как в [262], или с использованием метода k-ближайших соседей для выбора примеров в контексте [411]. В данном случае рассматривался тип задачи, известный заранее и заданный как GT.

Результаты экспериментов в задаче ALFRED

Были проведены две серии экспериментов в режимах оценки подзадач, генерации полного плана и пошаговой генерации для типов задач Pick&Place, Heat&Place и Stack&Place по отдельности с вычислением значений описанных выше метрик. В первом блоке экспериментов для каждого типа задачи индивидуально настраивался запрос одного типа, содержащей примеры только этого типа. Во втором блоке запрос включал примеры всех семи типов задач, чтобы определить, обладает ли модель способностью генерировать корректный сложный план и может ли она классифицировать тип задачи.

Для режима оценки подзадач были проверены GPT-J, GPT-NeoX и предложены по 35 задач каждого типа, случайным образом выбранных из набора данных ALFRED. Для каждой модели эксперименты проводились повторно с разной длиной запросов (содержащих разное количество примеров). Затем из полученных результатов были выбраны лучшие. Результаты приведены в табл. 2.

Таблица 2 — Результаты БЯМ в трех различных режимах планирования с запросами для конкретной задачи.

Task	GPT-J (6B)				GPT-NeoX (20B)				OPT (30B)			
	ACC	AEM	LCS	H	ACC	AEM	LCS	H	ACC	AEM	LCS	H
Subtask Evaluation Mode												
PP	0.57	1.00	0.85	0.89±0.00	0.63	1.00	0.87	0.97±0.00	0.66	1.00	0.87	0.85±0.02
HP	0.77	0.86	0.88	0.83±0.03	0.74	0.89	0.88	0.83±0.05	0.60	0.71	0.79	0.69±0.03
SP	0.31	0.91	0.74	0.69±0.13	0.26	1.00	0.70	0.70±0.01	0.34	1.00	0.69	0.64±0.02
Full Plan Generation												
PP	0.14	0.91	0.52	0.85±0.01	0.32	1.00	0.59	0.91±0.05	0.48	1.00	0.76	0.97±0.03
HP	0.38	0.97	0.65	0.55±0.02	0.29	0.49	0.50	0.59±0.17	0.22	0.58	0.39	0.84±0.05
SP	0.20	0.57	0.48	0.66±0.05	0.20	0.94	0.52	0.66±0.05	0.17	0.39	0.50	0.60±0.03
Step by Step Plan Generation												
PP	0.62	0.97	0.79	0.89±0.02	0.59	1.00	0.82	0.96±0.03	0.29	1.00	0.71	0.78±0.04
HP	0.40	0.60	0.60	0.47±0.03	0.34	0.51	0.57	0.40±0.03	0.29	0.74	0.61	0.60±0.08
SP	0.37	0.89	0.68	0.71±0.00	0.29	0.77	0.61	0.54±0.08	0.03	0.89	0.41	0.37±0.06

Поскольку в запросе приведены примеры только одного типа, последовательность действий успешно определена для каждой модели, о чем свидетельствует высокий показатель АЕМ. Существует значительный разрыв между показателями ACC и H из-за зашумленности набора данных ALFRED.

OPT превосходит две другие модели по эффективности выполнения двух типов задач, но уступает им по показателям H. Было обнаружено, что увеличение числа параметров модели (с GPT-J до OPT) не обеспечивает существенного повышения производительности, особенно в случае однотипных запросов. В целом, в отношении метрики H предложенный подход с GPT-NeoX в качестве БЯМ с однотипным запросом обеспечивает наилучшее качество примерно в 83%.

В проведенных экспериментах со смешанными запросами каждый из них содержит по три примера для каждого из семи типов задач общей длиной 21. Результаты приведены в табл. 3.

Таблица 3 — Результаты БЯМ в трех различных режимах планирования со смешанным запросом.

Task	GPT-J (6B)				GPT-NeoX (20B)				OPT (30B)			
	ACC	AEM	LCS	H	ACC	AEM	LCS	H	ACC	AEM	LCS	H
Subtask Evaluation Mode												
PP	0.43	0.60	0.71	0.56±0.09	0.46	0.54	0.68	0.54±0.03	0.03	0.03	0.49	0.26±0.39
HP	0.51	0.60	0.75	0.54±0.08	0.29	0.40	0.62	0.53±0.19	0.37	0.51	0.66	0.40±0.03
SP	0.06	0.26	0.59	0.17±0.03	0.11	0.80	0.61	0.42±0.11	0.03	0.74	0.57	0.19±0.06
Full Plan Generation												
PP	0.12	0.91	0.32	0.77±0.06	0.18	1.00	0.50	1.00±0.00	0.48	1.00	0.76	0.86±0.03
HP	0.14	0.14	0.40	0.70±0.01	0.20	0.51	0.43	0.70±0.06	0.22	0.58	0.39	0.62±0.11
SP	0.00	0.40	0.15	0.39±0.05	0.00	0.91	0.21	0.60±0.08	0.17	0.39	0.50	0.60±0.17
Step by Step Plan Generation												
PP	0.47	1.00	0.65	0.68±0.07	0.62	0.97	0.14	0.91±0.00	0.21	0.97	0.48	0.75±0.05
HP	0.37	0.54	0.61	0.48±0.02	0.43	0.46	0.64	0.55±0.05	0.00	0.00	0.16	0.41±0.08
SP	0.09	0.49	0.53	0.39±0.05	0.14	0.71	0.52	0.53±0.04	0.03	0.36	0.19	0.43±0.15

В режиме оценки подзадачи значения показателей ухудшились по сравнению с первым блоком экспериментов. Наиболее сложная задача типа SP имеет наименьшие значения ACC. Однако для GPT-NeoX AEM составляет около 80%, что означает, что модель хорошо обрабатывает семантику этой задачи по сравнению с меньшим GPT-J. GPT-J превосходит две другие модели по типу HP. OPT, имеющий наибольшее количество параметров, показал наихудшие результаты, «переобучившись» для конкретных типов задач, присутствующих в запросе.

В данном разделе диссертационного исследования было рассмотрено применение общедоступных БФМ для составления плана действий на концептуальном уровне архитектуры NSLP для воплощенного агента. Были рассмотрены три режима работы модели: 1) режим оценки подзадачи, 2) генерация полного авторегрессионного плана и 3) пошаговая генерация авторегрессионного плана. В проведенных исследованиях использовались модели с разным количеством параметров, при этом существенного увеличения значений метрик с увеличением количества параметров не наблюдается, а в некоторых случаях обнаруживается снижение.

В целом, режим оценки подзадачи работает лучше, чем два других подхода с запросом для конкретной задачи. Этот режим является наиболее ресурсоемким, поскольку требует параллельной оценки всех подзадач, доступных агенту. Режим генерации плана с полной авторегрессией представляется худшим среди рассмотренных вариантов. Основная проблема с этим режимом заключается в том, что подзадачи, полученные таким образом, могут быть недоступны для выполнения агентом. Режим генерации пошагового авторегрессионного плана занимает промежуточное положение с настройкой запроса для конкретной задачи, но превосходит другие режимы со смешанным запросом. Несмотря на то что для этого требуется

сопоставление полученных подзадач с пространством подзадач агента, эта процедура не так ресурсоемка, как параллельная оценка подзадач, и может быть реализована с предварительно вычисленными векторными представлениями подзадач. С увеличением количества действий и объектов количество подзадач также увеличивается комбинаторно. Хотя сопоставление сгенерированных подзадач с подзадачами, доступными агенту, можно рассматривать как неявную обратную связь со средой, в данном разделе рассматривалась генерация плана без обратной связи, и этот подход был взят за основу для дальнейшей работы при рассмотрении свойств архитектуры NSLP.

2.5 Выводы

Реализация системы управления воплощенным когнитивным агентом, действующим в динамической непредсказуемой среде, требует декомпозиции задачи генерации поведения на несколько уровней. Была предложена архитектура, реализующая декомпозицию верхнего уровня на стратегический, тактический и реактивные уровни по эффективному горизонту последовательности решений или, другими словами, по времени принятия решений — от максимального (часы) до минимального (секунды и микросекунды). В STRL раскрываются взаимосвязи между компонентами, реализующими перцептивный анализ ситуации (составление карты, сегментация объектов, трекинг объектов, генерация плана перемещений), управление движением (задачи стабилизации, следования по построенной траектории, парирования возмущений) и стратегическое планирование с использованием современных методов обучения и планирования задач, в том числе и в многоагентной постановке.

Стратегический уровень архитектуры STRL не обладал необходимой гибкостью и позволял реализовывать лишь короткие горизонты принятия решений. Для увеличения степени адаптивности в диссертации был более подробно проработан стратегий уровень генерации поведения, что привело к созданию нейросимвольной архитектуры планирования и обучения NSLP. В NSLP выделяется концептуальный (символьный) уровень принятия решений с использованием графового дискретного представления и методов генерации концептуального плана или последовательности подцелей на основе предобученных языковых моделей. На сенсорном уровне NSLP рассматривается обобщенная задача обучения с подкреплением на основе латентной модели среды с возможностью использования библиотеки предобученных навыков (получаемых в том числе и за счет предобучения в симуляционных средах). На сенсорном уровне проведен теоретический анализ эффективности использования объектной декомпозиции и модели среды для формирования субоптимальной стратегии.

Ключевым, интеграционным компонентом архитектуры NSLP является объектно-центричная нейросимвольная модель, которая позволяет переводить концептуальный план в верхнеуровневую стратегию. Последняя, в свою очередь, реализуется за счет библиотеки

навыков. Необходимость такого модели демонстрируется на классическом наборе данных ALPHRED, в котором только концептуальный уровень с использованием больших языковых моделей показывает не очень высокие результаты по эффективности составления плана действий.

Глава 3. Обучение и использование модели в архитектуре NSLP

В данной главе будут рассмотрены различные реализации процесса построения модели среды и ее использования при обучении стратегии в архитектуре NSLP. В качестве основного источника данных для построения аппроксимации модели среды (или *модели мира*) будут использоваться пополняемая память прецедентов или набор (пакет) траекторий. Основные результаты, представленные в данной главе, были опубликованы в [18; 57; 94] (раздел 3.1), [22; 28] (раздел 3.2), [2; 7; 10; 12; 67; 70; 77] (раздел 3.3), [87; 97] (раздел 3.4).

3.1 Модель среды в архитектуре «актор-критик»

В данном разделе диссертационного исследования рассмотрена реализация сенсорного уровня работы архитектуры NSLP без использования иерархии навыков (рисунок 3.1). Исследованы особенности организации работы актора и критика, использующие латентную модель динамики M_L , по которой генерируется так называемый латентный план. В разделе используется первичная сенсорная информация o_t , которая является результатом обработки получаемого агентом наблюдения без выделения объектов и связей между ними. Основные результаты, изложенные в данном разделе, представлены в публикациях [18; 57; 94].

Как было упомянуто в разделе 1.3, к настоящему времени разработаны подходы, которые используют обученную модель динамики как в исходном пространстве состояний [439], так и в абстрактном (латентном) пространстве [386; 419; 54] для ускорения обучения безмодельных алгоритмов на модельных траекториях, полученных семплированием из такой вероятностной модели среды. В другой группе алгоритмов [476; 613; 624] вместе с моделью среды одновременно строится модель функции полезности состояний. В этом случае модель среды и модель функции полезности состояний используются вместе для планирования поведения с помощью многошаговой оценки полезности действий. Для обучения модели состояний применяется Q-обучение.

В данном разделе рассматривается интеграция безмодельных алгоритмов семейства «актор-критик» и алгоритмов, использующих модель среды для оценки полезности действий. Предложено использовать Q-сеть с упреждающим поиском по дереву (TreeQN) [624], предсказывающую полезность действий, в качестве критика в алгоритме усредненного «актор-критика» (MAC) [427] и в алгоритме оптимизации ближайшей стратегии (PPO) [512].

Эксперименты с предлагаемыми алгоритмами проведены в среде Pong из популярного набора сред для оценки качества алгоритмов обучения с подкреплением Atari 2600 [609], а также в среде Crafter [308], характерной особенностью которой являются необходимость глубокого исследования и принятия решений в условиях редкого и отложенного вознаграждения. Эмпирические результаты показывают, что интеграция MAC и TreeQN

обучается быстрее, чем оригинальные алгоритмы MAC и TreeQN. Однако интеграции PPO и TreeQN демонстрирует ухудшение скорости обучения по сравнению с PPO и TreeQN. Такой результат может быть вызван достаточно медленным обучением модели функции полезности состояний из-за сложности предсказания скалярных значений полезности.

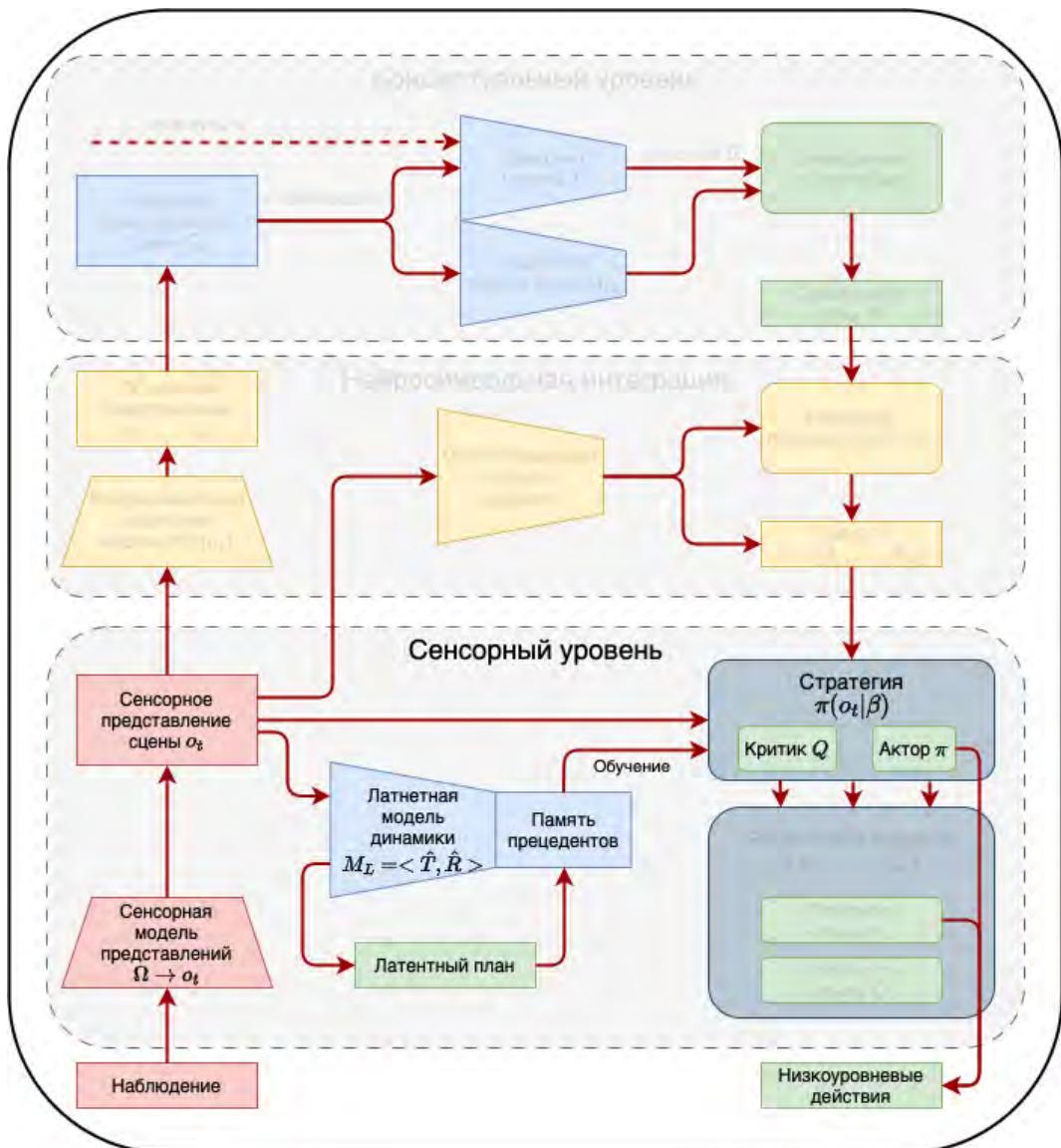


Рисунок 3.1 — Архитектура NSLP с выделенным сенсорными уровнем, на котором используется подход «актора-критика» для обновления латентной модели среды и плоской стратегии агента без использования навыков.

3.1.1 «Актор-критик» с моделью среды

Широко применяемым подходом к решению задач обучения с подкреплением являются методы семейства «актор-критик» [135; 512; 581], сочетающие в себе как построение стратегии, так и самой функции полезности. При этом компоненты,

представляющие стратегию и функцию полезности, явным образом разделены. Компонент, аппроксимирующий стратегию, отвечает за выбор действия и называется актором. Критик аппроксимирует функцию полезности, таким образом оценивая действия, выбранные актором. В общем виде функция потерь модели актора с параметрами θ и энтропийной регуляризацией имеет вид:

$$\begin{cases} L_{actor}(\theta) = \log [\pi_\theta(s,a)] F, \\ \theta^* = \arg \max_\theta L_{actor}(\theta) + \alpha H(\pi_\theta(\cdot|s_t)), \end{cases}$$

где $H(\pi_\theta(\cdot|s_t))$ — энтропия стратегии, $\alpha > 0$ — коэффициент, контролирующий вклад энтропии в функцию потерь. В качестве F может применяться оценка полезности действий $Q(s,a)$, TD-ошибки или функции преимущества $A(s,a)$. Критик с параметрами η обучается с использованием квадратичной функции потерь:

$$\begin{cases} L_{critic}(\eta) = -(G^\lambda - Critic_\eta(s,a))^2, \\ \eta^* = \arg \max_\eta L_{critic}(\eta), \end{cases}$$

где G^λ — λ -отдача, а $Critic_\eta(s,a)$ — выход критика. Модели актора и критика обучаются итеративно на подвыборках эпизодов взаимодействия со средой, причем параметры моделей обновляются после каждой итерации.

Во многих практических задачах функции перехода и вознаграждения неизвестны, а агенту доступно лишь значение вознаграждения r_t , полученного после применения действия a_t в текущем состоянии s_t . В подходах безмодельного обучения оптимальная стратегия строится напрямую на основе данных, приобретенных при взаимодействии агента со средой. С другой стороны, на основе опыта взаимодействия со средой агент может построить модель среды, аппроксимирующую функцию переходов $P \approx p$ и функцию вознаграждения $R \approx r$, и применить её предсказания в качестве дополнительного сигнала для обучения стратегии. Схема обучения с использованием модели среды показана на рисунке 1. При этом точность модели среды имеет решающее значение для поиска оптимальной стратегии: безошибочная модель позволяет получить оптимальную стратегию без взаимодействия агента со средой, однако требует большого количества шагов для глубокого исследования среды и сбора данных для обучения модели. Менее точную модель получить проще, но ошибочные предсказания ухудшают качество получаемой стратегии особенно при использовании многошаговых прогнозов, когда ошибки модели накапливаются.

Одним из возможных способов улучшения качества алгоритмов «актор-критик» является уточнение предсказаний критика. В этой работе мы используем в качестве критика вариант модели TreeQN, которая задействует модель динамики среды для построения функции полезности действий.

Нейросетевая модель TreeQN обладает рекурсивной структурой и используется для решения задач обучения с подкреплением с дискретным набором действий. TreeQN динамически строит дерево, рекурсивно применяя модель переходов в абстрактном пространстве состояний, и агрегирует предсказываемую награду и полезность состояний для оценки функции полезности действий. Схема работы алгоритма TreeQN изображена на Рисунке 3.2. Основными компонентами TreeQN являются:

- Кодировщик $encoder : S \rightarrow Z$, служащий для получения абстрактного представления состояний среды;
- Модель функции переходов $P : Z \times A \rightarrow Z$ в абстрактном пространстве состояний;
- Модель функции вознаграждения $R : Z \times A \rightarrow \mathbb{R}$;
- Модель функции полезности состояний $V : Z \rightarrow \mathbb{R}$.

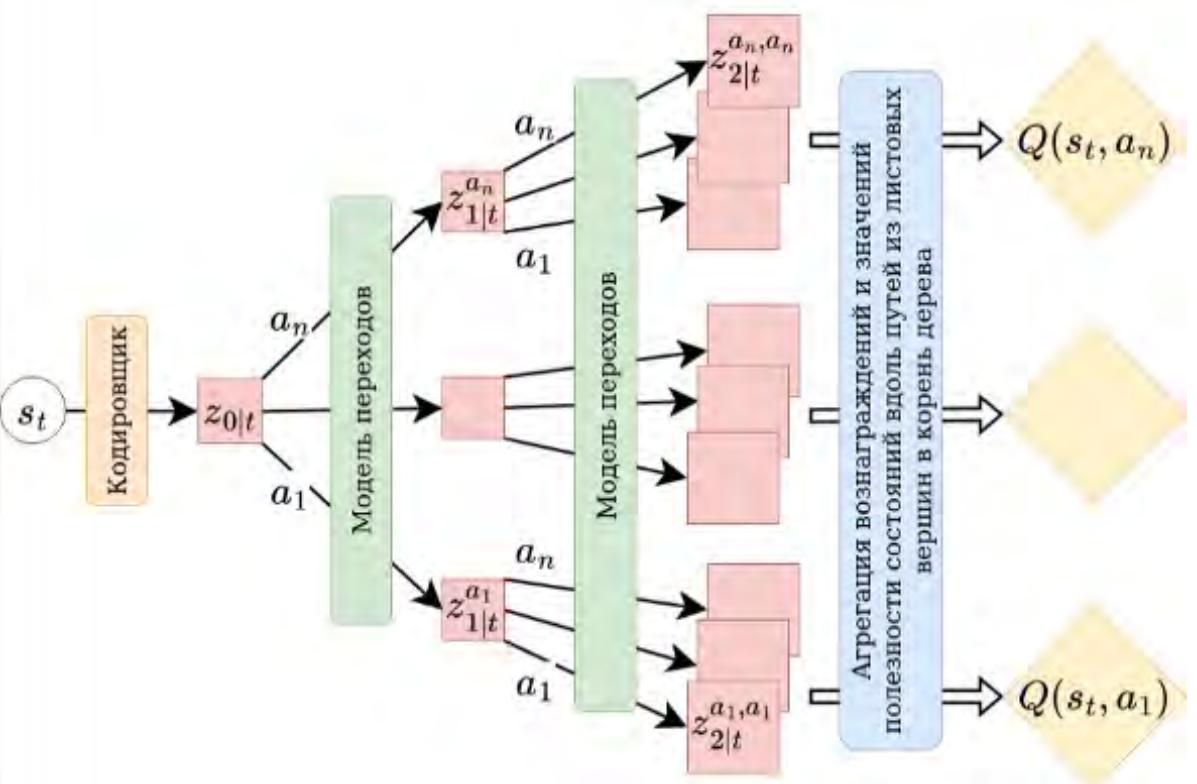


Рисунок 3.2 — Схема расчета полезности действий в алгоритме TreeQN с глубиной $d = 2$.

Для текущего состояния среды $s_t \in S$ вычисляется его абстрактное представление $z_{0|t} = encoder(s_t)$, и путем рекурсивного применения модели переходов P динамически строится дерево последующих состояний $z_{l+1|t}^{a_i} = P(z_{l|t}, a_i)$ фиксированной глубины d для всех действий $a_i \in A$, где t — индекс шага в среде и $l < d$ — индекс шага планирования. Функция полезности действий в состоянии s_t оценивается как функция полезности корня дерева $Q^{TreeQN}(s_t, a) = Q^0(z_{0|t}, a)$ с использованием алгоритма TD(λ) и k-шаговых отдач, рассчитанных по дереву:

$$\begin{aligned} Q^l(z_{l|t}, a_i) &= r(z_{l|t}, a_i) + \gamma V^{(\lambda)}(z_{l+1|t}), \\ V^{(\lambda)}(z_{l|t}) &= \begin{cases} V(z_{l|t}^{a_i}) & l = d \\ (1 - \lambda)V(z_{l|t}^{a_i}) + \lambda b(Q^{l+1}(z_{l+1|t}^{a_i}, a_j)) & l < d \end{cases}, \\ b(\mathbf{x}) &= \sum_i x_i softmax(\mathbf{x})_i \end{aligned}$$

Модель R выдает вознаграждения $r_{l|t}^a = R(z_{l|t}, a)$ вдоль путей из корня дерева в листовые вершины, а полезность состояний $z_{l|t}$ оценивается как усреднение предсказания модели $V(z_{l|t})$ с весом $1 - \lambda$ и softmax-агрегации полезности действий последующих состояний $z_{l+1|t}^{a_i}$ с весом λ . В качестве функции потерь используется квадрат отклонения предсказанных моделью значений полезности действий от n-шаговой отдачи в среде с применением замороженной версии сети для бутстрэпа.

3.1.2 Поиск по дереву модели в критике

Предлагаемая интеграция обучения и планирования в данном случае заключается в использовании сети TreeQN в качестве критика. В качестве алгоритмов «актор-критик» для интеграции с TreeQN рассматриваются PPO и MAC.

Алгоритм PPO входит в семейство методов «актор-критик». Основная идея алгоритма заключается в том, чтобы обновление стратегии во время обучения не приводило к ухудшению её качества. Актор аппроксимирует стратегию π_θ , критик — функцию полезности состояний V_η , где θ — параметры актора и η — параметры критика. Функции потерь для версии PPO с усечением в целевой функции актора имеют вид:

$$\begin{cases} L_{actor}(\theta) = \min [\rho_t(\theta) A_t, \text{clip}(\rho_t(\theta), 1 - \varepsilon, 1 + \varepsilon) A_t], \\ L_{critic}(s_t, \eta) = -(G^\lambda(s_t) - V_\eta(s_t))^2, \end{cases}$$

где t — индекс шага в среде; $G^\lambda(s_t)$ — λ -отдача; ρ_t — отношение вероятностей действий между новой и старой стратегиями; ε — параметр, контролирующий степень изменения стратегии на одном обновлении. Выход критика используется в обобщенной оценке функции преимущества в качестве предсказания базового уровня: $A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \dots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \delta_t = r_t + \gamma V_\eta(s_{t+1}) - V_\eta(s_t)$.

Поскольку в PPO критик применяется для оценки полезности состояний, модель TreeQN необходимо преобразовать в модуль, предсказывающий функцию полезности состояний. Для этого выдаваемые TreeQN значения полезности действий усредняются по текущей стратегии π_θ . При этом функция потерь критика не изменяется:

$$V_\eta^{TreeQN}(s_t) = \sum_{a \in A} \pi_\theta(a|s_t) Q_\eta^{TreeQN}(s_t, a).$$

Предполагая, что за счет упреждающего поиска на несколько шагов вперед TreeQN будет предсказывать базовый уровень более точно, чем многослойный персепtron, который обычно используется для реализации критика в PPO, можно ожидать, что гибридный алгоритм PPO+TreeQN продемонстрирует качественный прирост.

Алгоритм MAC разработан для сред с дискретным пространством действий. Особенностью, которая отличает его от других алгоритмов семейства «актор-критик», является то, что для оценки градиента стратегии используются оценки полезности всех действий, а не только тех, которые были выполнены при взаимодействии со средой. Актор аппроксимирует стратегию π_θ , критик — функцию полезности действий Q_η , где θ — параметры актора и η — параметры критика. Функции потерь для актора и критика:

$$\begin{cases} L_{actor}(\theta) = \sum_{a \in A} \pi_\theta(a|s_t) Q_\eta(s_t, a), \\ L_{critic}(\eta) = -(G^\lambda(s_t, a_t) - Q_\eta(s_t, a_t))^2, \end{cases}$$

где t — индекс шага в среде; $G^\lambda(s_t, a_t)$ — λ -отдача.

Выходом критика является оценка значений функции полезности действий, которые подставляются в градиент стратегии. Таким образом можно напрямую задействовать TreeQN

в качестве критика, не внося никаких изменений ни в MAC, ни в TreeQN. Как и в случае с PPO, мы ожидаем, что многошаговое прогнозирование уточнит предсказание функции полезности действий и ускорит обучение стратегии в гибридном алгоритме MAC+TreeQN по сравнению с оригинальной реализацией MAC, где используется многослойный персепtron.

3.1.3 Экспериментальное исследование актор-критиков с моделью среды

Работа гибридных алгоритмов PPO+TreeQN и MAC+TreeQN оценивалась в игровых средах Atari 2600 Pong и Crafter. Эксперименты были проведены с целью сравнения качества гибридных алгоритмов с оригинальными реализациями PPO и MAC, а также с такими алгоритмами, используемыми в качестве базового уровня в современных задачах, как безмодельные Rainbow [518] и алгоритм асинхронной оптимизации ближайшей стратегии (APPO) [544] и использующие модель среды DreamerV2 [419] и SimPLe [439].

Реализация гибридного алгоритма PPO+TreeQN основана на библиотеке Stable Baselines v3 [591], а интеграция MAC+TreeQN использует собственную реализацию алгоритма MAC. Также для сравнения был реализован вариант PPO (large critic), у которого в качестве критика выступает четырехслойный персепtron с количеством параметров, равным количеству параметров TreeQN.

Crafter [308] представляет собой двухмерную видеоигру, упрощенную версию Minecraft [5]. Целью агента является выживание в случайно генерированном мире. Для этого агенту необходимо добывать пищу и воду, находить убежище для сна, защищаться от монстров, собирать ресурсы и создавать инструменты. Карта мира — 64-клеточная квадратная область, каждая ячейка которой соответствует определенному типу местности. В поле зрения агента входит лишь 3 ячейки по вертикали и 4 по горизонтали. У агента есть уровни еды, воды и отдыха, которые снижаются со временем и восстанавливаются за счет питья из озера, сбора пищи и сна. Как только один из трех уровней достигает нуля, игрок начинает терять очки здоровья. Он также может терять очки здоровья при нападении зомби и скелетов. Когда очки здоровья достигают нуля, игрок умирает. Агент может собирать ресурсы и делать из них оружие или инструменты. Для сбора некоторых ресурсов необходимо иметь инструменты, произведенные из других ресурсов. Например, для сбора камня агент должен обладать киркой, сделанной из дерева.

Наблюдения Среда выдаёт цветное изображение размером 64x64. Изображение включает часть карты, которая попадает в поле зрения агента, а также содержимое инвентаря и текущие показатели уровня здоровья, пищи, воды и отдыха. Пример изображения показан на рисунке 3.3.

Действия Пространство действий состоит из 17 элементов: передвижение в четырех направлениях, переход в режим сна, размещение объекта из инвентаря перед собой,

создание инструмента, а также по одному действию на применение каждого типа объекта и инструмента.

Окончание эпизода Эпизод заканчивается, если уровень здоровья игрока опустился до нуля или игрок совершил 10000 шагов.

Вознаграждение В среде определены $N = 22$ достижения, соответствующие различным способностям агента: сбор различных ресурсов, пищи, воды, создание инструментов, уничтожение монстров и т.д. Некоторые достижения зависят от других; так образуется иерархия достижений. Агент получает награду +1 каждый раз, когда впервые выполняет достижение. Также он получает награду -0.1 за потерю одного очка здоровья и +0.1 за его восстановление. Таким образом, наличие иерархии достижений требует от агента глубокого исследования среды и принятия решений в условиях редкого и отложенного вознаграждения.

Оценка эффективности агента В качестве меры разнообразия способностей, выученных агентом, выбрана частота выполнения достижений [308]: для каждого достижения вычисляется отношение количества эпизодов, в которых агент выполнил это достижение, к общему количеству эпизодов. Частоты выполнения всех достижений $\{f_i\}_{i=1}^{i=N}$ агрегируются в метрику score с помощью геометрического усреднения: $score = \exp\left(\frac{1}{N} \sum_{i=1}^N \ln(1 + f_i)\right) - 1$, где $f_i \in [0; 100]$ и $N=22$.

В среде Pong проводились эксперименты со следующими вариантами алгоритмов: PPO+TreeQN: $d=2$ — интеграция PPO с TreeQN глубиной $d=2$; PPO+TreeQN: $d=3$ — интеграция PPO с TreeQN глубиной $d=3$; PPO (stable-baselines3) — реализация PPO в библиотеки stable-baselines3; MAC+TreeQN: $d=2$ — интеграция MAC с TreeQN глубиной $d=2$; MAC+TreeQN: Q-normalization — вариант интеграции MAC с TreeQN глубиной $d=2$ с нормализацией значений функции полезности действий; MAC+TreeQN: no-target-critic — вариант интеграции MAC с TreeQN глубиной $d=2$, в которой не используется замороженная сеть TreeQN для бутстрэпа; MAC — реализация оригинального алгоритма MAC.

Графики скользящего среднего суммарной награды за эпизод представлены на рисунке 3.4. Гибридные модели PPO+TreeQN обучаются медленнее, чем оригинальный алгоритм PPO (stable-baselines3), причем увеличение глубины предсказания не оказывается на качестве. С другой стороны, применение TreeQN в алгоритме MAC привело к значительному ускорению обучения. Дополнительный небольшой прирост удалось получить при отказе от использования замороженной версии критика для расчёта бутстрэп-оценки отдачи. Однако обучение модели MAC+TreeQN происходит все еще значительно медленнее, чем PPO. Также в случае MAC+TreeQN заметный эффект дает применение нормализации значений функции полезности действий по подвыборке опыта взаимодействия со средой, которая используется на данной итерации обучения: на раннем этапе до 1.5 млн шагов обучение ускоряется, но затем скорость обучения существенно падает. При этом значительно растет дисперсия между запусками алгоритма.

В среде Crafter проводились эксперименты со следующими вариантами алгоритмов: DreamerV2 — оригинальная реализация алгоритма DreamerV2; Rainbow — оригинальная реализация алгоритма Rainbow; PPO+TreeQN: $d=2$ — интеграция PPO с TreeQN глубиной

$d=2$; PPO (stable-baselines3) — реализация PPO в библиотеки stable-baselines3; TreeQN: $d=2$ — оригинальная реализация TreeQN глубиной $d=2$; PPO (large critic) — вариант PPO, у которого в качестве критика выступает четырёхслойный персепtron с количеством параметров, равным количеству параметров TreeQN.; APPO — оригинальная реализация алгоритма APPO; SimPLe — оригинальная реализация алгоритма SimPLe; MAC+TreeQN — интеграция MAC с TreeQN глубиной $d=2$.



Рисунок 3.3 — Наблюдение в среде Crafter. В центре расположен игровой персонаж, внизу размещен инвентарь. В поле зрения находятся различные типы местности (трава, вода) и деревья. Агент может съесть корову, которая движется справа, для пополнения уровня пищи.

Графики метрики *score* представлены на рисунках 3.5 и 3.6, скользящего среднего суммарной награды за эпизод — на рисунках 3.7 и 3.8. Стоит отметить, что ранжирование алгоритмов по обеим метрикам совпадает. Наименьшую скорость обучения показал алгоритм SimPLe, что, предположительно, связано с тем, что он оптимизирован для подмножества сред Atari в режиме малого взаимодействия со средой (не более 100000 шагов) и ему не хватает мощности для работы в такой сложной среде как Crafter.

В среде Crafter гибридный алгоритм PPO+TreeQN превосходит оригинальный PPO (stable-baselines3), однако уступает варианту PPO (large critic), у которого критик имеет столько же параметров, сколько и TreeQN, но состоит только из четырех полно связных слоев. TreeQN и интеграция MAC+TreeQN уступают PPO на начальном этапе до 4.5 млн шагов, но

на 10 млн шагов уже опережают все варианты PPO и PPO+TreeQN. На отрезке до 120 млн шагов MAC+TreeQN демонстрирует более эффективное обучение, чем TreeQN. Медленнее всего обучается APPO, но к 100 млн шагов он уже превосходит все остальные алгоритмы. На начальном этапе до 4.5 млн шагов самое быстрое обучение демонстрируют Rainbow и DreamerV2, причём DreamerV2 значительно превосходит все остальные алгоритмы, а APPO достигает таких результатов только к 70 млн шагов.

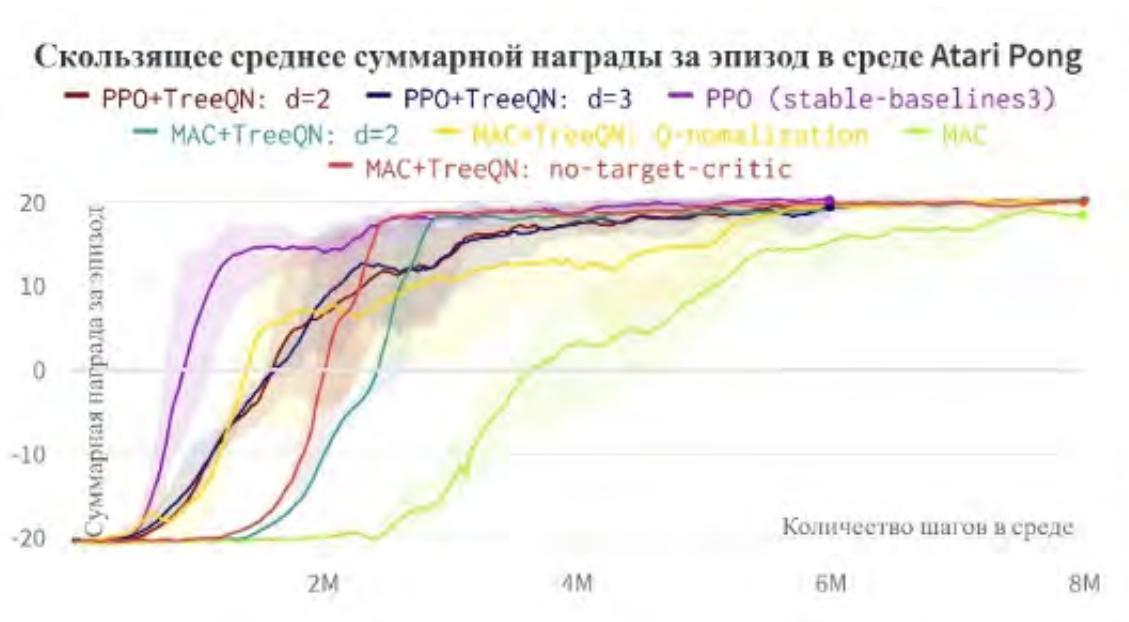


Рисунок 3.4 — График зависимости скользящего среднего суммарного вознаграждения за эпизод от количества шагов в среде Atari Pong. Произведено усреднение кривых по трем запускам алгоритмов. Затенение показывает минимальное и максимальное значение в группе запусков.

В работе представлены гибридные алгоритмы PPO+TreeQN и MAC+TreeQN, комбинирующий безмодельных алгоритмы семейства «актор-критик» и алгоритм, применяющий модель среды для оценки полезности действий. Эксперименты на средах Atari 2600 Pong и Crafter показали, что интеграция TreeQN в качестве критика в алгоритм MAC ускоряет обучение стратегии, а в случае интеграции с PPO происходит замедление обучения.

Гибридный алгоритм PPO+TreeQN демонстрирует ухудшение скорости обучения по сравнению с оригинальным алгоритмом PPO (stable-baselines3) в среде Pong и его вариантом PPO (large critic) в среде Crafter. Предположительно, это связано с медленным обучением TreeQN из-за сложности предсказания скалярных значений функции полезности. Поэтому усложнение критика в алгоритме PPO, который в оригинальной версии уже достаточно эффективен, лишь замедлило обучение.

С другой стороны, интеграция MAC+TreeQN обучается быстрее, чем оригинальные MAC и TreeQN. Можно сделать вывод, что суммирование оценок функции полезности действий при расчете градиента стратегии в алгоритме MAC уменьшает влияние ошибок модели TreeQN. Также стоит отметить, что в среде Crafter алгоритм MAC+TreeQN

превосходит APPo на первом этапе обучения (до 45 млн шагов по метрике и до 20 млн по суммарной награде за эпизод) и уступает лишь алгоритмам DreamerV2 и Rainbow.

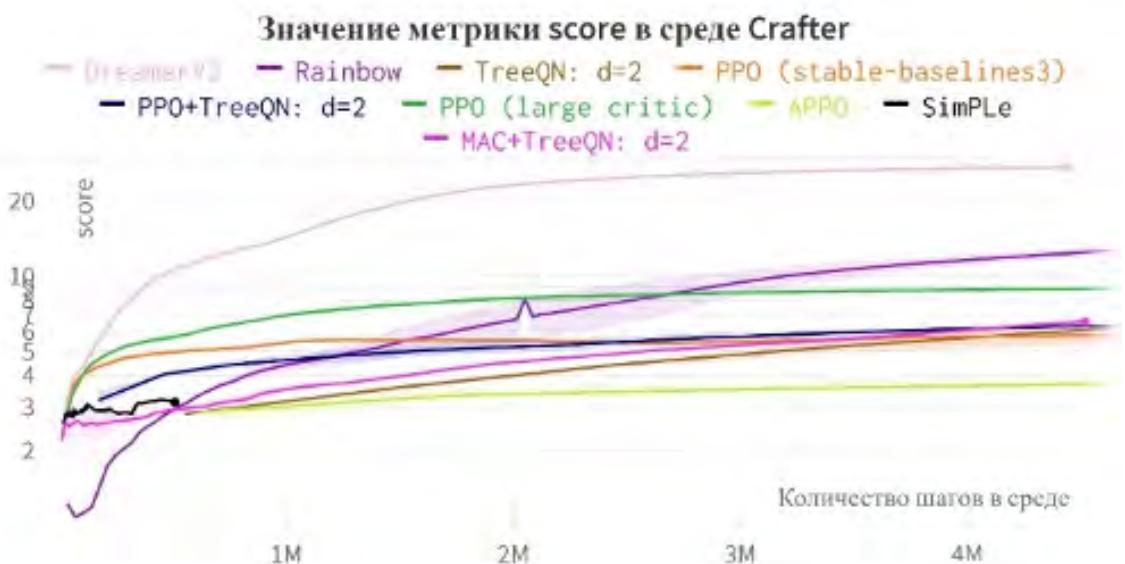


Рисунок 3.5 — Графики зависимости метрики score от количества шагов в среде Crafter для всех рассматриваемых алгоритмов с бюджетом взаимодействия со средой не более 4.5 млн шагов. Для оси ординат применен логарифмический масштаб. Произведено усреднение кривых по трем запускам алгоритмов. Затенение показывает минимальное и максимальное значение в группе запусков.



Рисунок 3.6 — Графики зависимости метрики score от количества шагов в среде Crafter для медленно обучающихся алгоритмов с бюджетом взаимодействия со средой не более 120 млн шагов. Произведено усреднение кривых по трем запускам алгоритмов. Затенение показывает минимальное и максимальное значение в группе запусков.

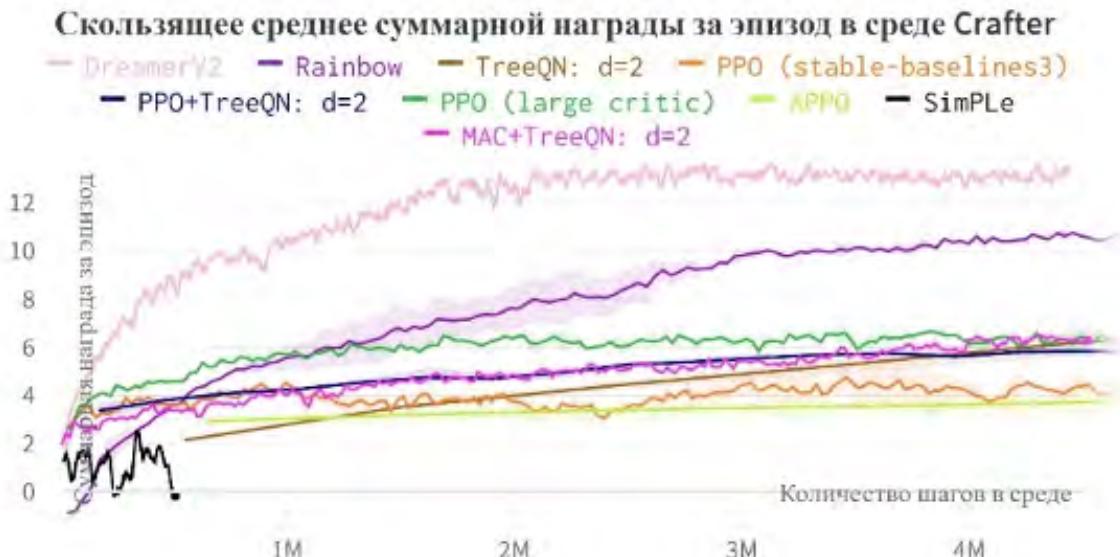


Рисунок 3.7 — Графики зависимости скользящего среднего суммарного вознаграждения за эпизод от количества шагов в среде Crafter для всех рассматриваемых алгоритмов с бюджетом взаимодействия со средой не более 4.5 млн шагов. Произведено усреднение кривых по трем запускам алгоритмов. Затенение показывает минимальное и максимальное значение в группе запусков.

Скользящее среднее суммарной награды за эпизод в среде Crafter

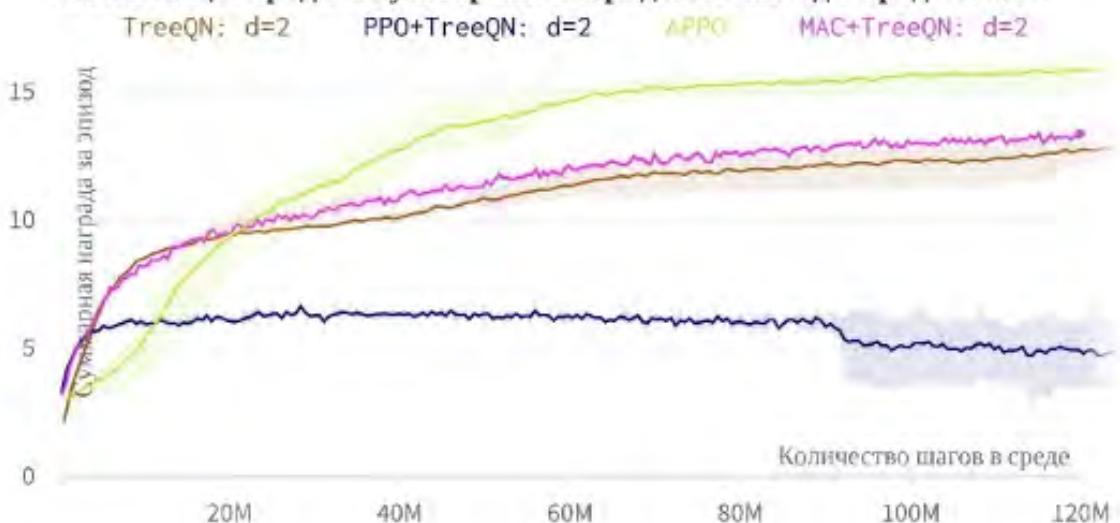


Рисунок 3.8 — Графики зависимости скользящего среднего суммарного вознаграждения за эпизод от количества шагов в среде Crafter для медленно обучающихся алгоритмов с бюджетом взаимодействия со средой не более 120 млн шагов. Произведено усреднение кривых по трем запускам алгоритмов. Затенение показывает минимальное и максимальное значение в группе запусков.

3.2 Реализация модели в задачах управления

В данном разделе диссертационного исследования продолжается рассмотрение реализации сенсорного уровня работы архитектуры NSLP без использования иерархии

навыков (рисунок 3.1). Исследованы особенности построения эффективной модели среды M_L для задач управления. В разделе также используется первичная сенсорная информация o_t , которая является результатом обработки получаемого агентом наблюдения без выделения объектов и связей между ними. Основные результаты, изложенные в данном разделе, представлены в публикациях [22; 28].

Задачи робототехнической манипуляции часто связаны с неопределенностью и недетерминированными причинно-следственными связями внутри среды. Модели динамики манипуляторов в детерминированных и предсказуемых условиях хорошо изучены. В этих условиях эффективны классические методы управления. Однако коллаборативным манипуляторам часто приходится оперировать различными объектами именно в слабо предсказуемой среде с неизвестными параметрами. Управление таким воплощенным робототехническим агентом обычно строится на основе стратегий, полученных с помощью глубокого обучения с подкреплением, что позволяет находить последовательность действий с высоким качеством управления по данным из среды [516; 551]. Однако управление многозвездными манипуляторами является сложной задачей и в обучении с подкреплением из-за большого числа степеней свободы в системе, высоких требований к точности и необходимости строить модель системы в непрерывном времени [197]. В данном разделе рассматривается подход на основе нейронных обыкновенных дифференциальных уравнений (NODE) [463] в контексте архитектуры NSLP, который позволяет преодолеть эти проблемы, добавив модель среды в непрерывном времени к методу управления на основе семейства алгоритмов «актор-критик».

Существует несколько потенциальных преимуществ использования NODE для моделирования динамики среды при оптимизации стратегии на основе модели [304; 386; 652]. Во-первых, такая архитектура может обеспечить более гибкую и выразительную модель динамики в непрерывном времени [238] и позволяет описывать траекторию как непрерывную функцию, которая может лучше отражать сложные взаимосвязи процессов в среде. Во-вторых, ее можно использовать для моделирования динамики в различных временных масштабах [460; 540]. Традиционные модели с дискретным временем, например, основанные на рекуррентных нейронных сетях, требуют дискретизации временного шага на интервалах фиксированного размера, что может привести к ограничениям в моделировании динамики в различных временных масштабах. Было показано, что архитектура NODE обеспечивает высокое качество при решении различных задач, включая классификацию изображений и прогнозирование временных рядов [463]. Это говорит о том, что использование данного алгоритма может быть многообещающим подходом к моделированию динамики среды в методах обучения с подкреплением на основе модели среды.

В данном разделе диссертационного исследования предложен новый метод обучения стратегии, в котором безмодельный алгоритм обучается на выборках из модели мира, представленной NODE. Метод тестируется при решении сложной задачи открывания двери роботом. Во многих предыдущих работах NODE исследовался в различных постановках задач, но лишь немногие рассматривали задачу моделирования динамики, зависящей от действий агента [125; 238]. Помимо оригинальной реализации одновременного обучения

и планирования в архитектуре NSLP, полученные результаты вносят вклад в методы интеллектуального управления, заключающийся, во-первых, в использовании NODE для моделирования динамики системы в непрерывном времени с учетом управляющих входов и, во-вторых, в применении модели динамики, основанной на NODE, в задаче обучения с подкреплением.

3.2.1 Использование нейросетевых аппроксиматоров дифференциальных уравнений в качестве модели динамики

Оптимизация стратегии на основе модели — это особый подход к обучению с подкреплением на основе модели, который предполагает регуляризацию для поддержки монотонной оптимизации стратегии агента с использованием модели среды. Недавние достижения в методах оптимизации стратегии на основе моделей среды были сосредоточены на повышении точности и эффективности моделей динамики, а также на разработке новых алгоритмов, которые могут использовать эти модели для повышения эффективности выборок из среды и качества предсказаний. Например, в [462] используется управление на основе прогнозирующих моделей (MPC) в задаче обучения с подкреплением на основе модели, а в [386] обучается модель мира, которая используется для планирования при генерации новых выборок для обучения.

В [646] авторы предлагают алгоритм MBPO, который строит модель среды для оптимизации стратегии, чередуя сбор данных из среды и оптимизацию стратегии с использованием обученной модели. Было показано, что данный алгоритм обеспечивает высокую эффективность при решении различных задач непрерывного управления. В работах [237; 419] обучается одношаговая стратегия с использованием модели динамики среды, представленной рекуррентной нейронной сетью (RNN).

Использование структуры NODE для построения предсказательных моделей динамики вызвало значительный интерес у исследователей после первой работы по нейронным обыкновенным дифференциальным уравнениями [463]. Этот алгоритм использовался для построения модели динамики обучения с подкреплением на основе модели среды, как, например, в [238], включая работу в задаче с непрерывным временем [652]. Ряд работ посвящен моделированию динамики механических систем [300; 660]. Моделирование динамики переходов в среде, обусловленной действием агента, также изучалось в [125].

В качестве примера воплощения когнитивного агента в данном разделе рассматривается робототехнический манипулятор в задаче автоматического открытия двери для осуществления мобильной навигации внутри помещений. Возможность открывать стандартные двери в жилых и офисных помещениях полезна для эффективной навигации роботизированных систем в человеко-ориентированных средах. Дверь и робота можно рассматривать вместе как сложную систему. Кинематика и динамика такой системы

представлена моделью, включающей прямую кинематику робота и обратную кинематику двери. Эта модель также может включать неизвестные параметры (например, размер двери). Это свойство делает открытие двери теоретически интересной задачей для оценки работоспособности подходов одновременного обучения и планирования в NSLP. Этот же класс задач был, например, включен в набор задач для оценки качества работы методов мета-обучения [430].

Можно выделить следующие важные работы, применяющие обучение с подкреплением для этой задачи. В [215] применяется распределенный алгоритм нормализованной функции преимущества для обучения нескольких реальных роботов решению задачи открытия дверей. Еще одна работа, посвященная реальному роботу с обучаемой стратегией для открытия дверей, – это [224], где для решения данной задачи потребовалось 17,5 часов обучения в реальном мире. В некоторых других исследованиях рассматривается обучение задаче открытия дверей в симуляторе. В [250] предлагается сфокусироваться на обучении стратегии с использованием изображений в качестве наблюдений и без знания модели вознаграждения. Представлены эксперименты по задаче открытия дверей в среде MetaWorld [430], тогда как для экспериментов в реальном мире были выбраны другие задачи. В [217] применяется генетический алгоритм для поиска подходящих параметров алгоритма глубокого детерминированного градиента стратегии (DDPG). В указанных работах рассмотрены безмодельные методы обучения для открывания дверей. В качестве примера использования обучения на основе моделей можно упомянуть [196], которые рассматривают открытие двери как последовательность отдельных подзадач (приближение к ручке, вращение ручки и т.д.), где каждая подзадача имеет свою независимую модель динамики. Целью обучения общей модели было определение подходящих векторов для различных подзадач. В рамках предлагаемого в данном разделе подхода предлагается рассмотреть открытие двери как комплексную задачу и найти модель ее непрерывной динамики.

3.2.2 Использование NODE в качестве модели в цикле обучения

В рамках предлагаемого подхода к одновременному обучению и планированию используется схема оптимизации стратегии на основе модели, аналогичная [646], чтобы полностью обучать стратегию на синтетических выборках из модели среды, одновременно оптимизируя и стратегию и модель среды без предобучения (см. рисунок 3.9).

Для обучения стратегии используется алгоритм SAC [581]. Стратегия π обновляется на данных из памяти прецедентов (буфера) B_π , в котором хранятся только прецеденты, выбранные из модели среды E (см. рисунок 3.10). Чтобы сгенерировать траектории из модели среды, она переводится в исходное состояние и генерируется короткая траектория фиксированной длины, используя текущую стратегию. Распределение начальных состояний — это равномерное распределение по части памяти фиксированного размера B_E ,

которая содержит самые последние переходы, наблюдаемые в окружающей среде. Чтобы вычислить вознаграждение за предсказанные переходы, используется истинная функция вознаграждения для каждого окружения, в котором обучается агент.

Algorithm 1 General model-based policy optimization

```

1: Initialize policy  $\pi$ , dynamics model  $p_\theta$ , buffer for environment samples  $B_E$ , buffer
   for model samples  $B_\pi$ 
2: for  $N$  epochs do
3:   sample  $n$  transitions from environment under  $\pi$ ; add to  $B_E$ 
4:   Train model  $p_\theta$  on  $B_E$ 
5:   for  $M$  model rollouts do
6:     Sample  $o_t$  uniformly from the last  $q$  records in  $B_E$ 
7:     Perform  $k$ -step model rollout starting from  $s_t = g(o_t)$  using policy  $\pi$ ; add observed
       transitions to  $B_\pi$ 
8:   end for
9:   for  $G$  gradient updates do
10:    Update policy parameters on batch from  $B_\pi$ 
11:   end for
12: end for

```

Рисунок 3.9 — Общий алгоритм оптимизации стратегии с помощью модели.

Чтобы успешно обучать несколько взаимозависимых моделей, важно сбалансировать их частоты генерации выборок (семплирования) и обновления их параметров. Данные частоты фиксируются в качестве гиперпараметров. Модель динамики окружающей среды строится с использованием системы нейросетевых дифференциальных уравнений (NODE), обусловленной действиями агента. Данная система состоит из многослойного перцептрона (MLP) f_θ , наблюдателя g и решателя дифференциальных уравнений:

$$s' = f_\theta(s, a), s = g(o).$$

MLP учится предсказывать производную состояния s' при условии нахождения в состоянии s и выполнения действия a . Решатель дифференциальных уравнений интегрирует f_θ для получения траектории из начального состояния s_0 с учетом выбранной агентом последовательности действий.

Наблюдатель g реализован с помощью функций, определенных вручную, которые используют подмножество признаков наблюдения агента для восстановления состояния s на основе векторного наблюдения o . Для упрощения задачи предсказания предлагается использовать такой решатель путем построения исчерпывающие описательных, минимальных и интерпретируемых представлений s отдельно для каждой среды, в которой происходит обучение агента. Для интерполяции последовательности действий во время интегрирования динамики состояния используется константная интерполяция.

Модель обучается с использованием оптимизатора RMSprop путем минимизации среднеквадратичной ошибки MSE между прогнозируемыми и наблюдаемыми траекториями. Все траектории, наблюдаемые из окружающей среды, сохраняются в памяти прецедентов B_E . Затем равномерно выбираются наборы подпоследовательностей фиксированной длины

из этой памяти, чтобы на них обновить модель. Поскольку эти подпоследовательности, как правило, короткие, сопряженный метод не используется, а градиент вычисляется путем автоматического дифференцирования.

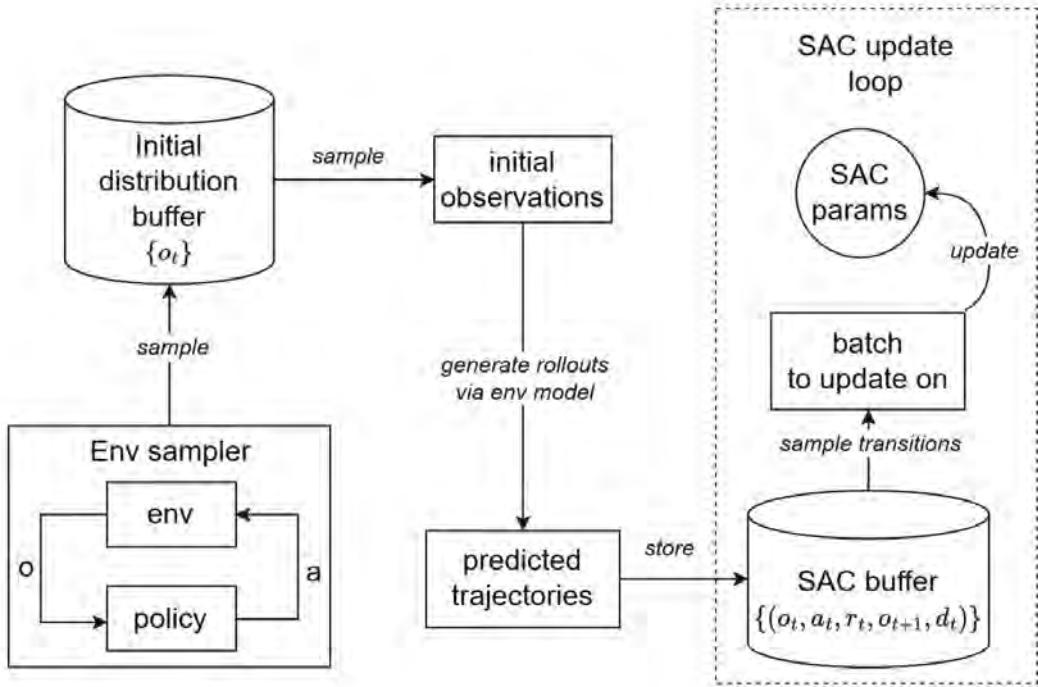


Рисунок 3.10 — Схема генерации выборок (семплирования) в процессе обучения. Здесь buffer — это память прецедентов.

Предлагаемый метод отличается от тех, которые задействуются в известных работах в этой области, выбором модели динамики и процессом обучения. В данном случае используем NODE, обусловленную на действия для моделирования динамики окружающей среды. Вместо использования кодировщика наблюдений в виде изображений предлагается восстанавливать векторные состояния динамической модели на основе наблюдений с помощью функции g , закодированной с использованием набор универсальных для конкретной среды правил и не имеющей обучаемых параметров. Эта функция формирует информационное состояние s , по которому легко определить его свою собственную производную, что в свою очередь более удобно для использования в цикле предсказания.

3.2.3 Экспериментальное исследование модели NODE

Экспериментальное исследование проведено путем обучения агента в различных средах и валидации полученной стратегии. Используемые среды основаны на *Reacher-v2* в библиотеке Gym [481], *Cheetah-run* в библиотеке DMC [232] и *DoorOpen* в библиотеке Metaword [430]. Все рассматриваемые среды построены на основе симулятора *MuJoCo*. Используемая версия *Reacher* позволяет осуществлять управление по углам сочленений

для облегчения поиска модели среды, а *DoorOpen* использует модифицированную функцию вознаграждения для облегчения поиска стратегии.

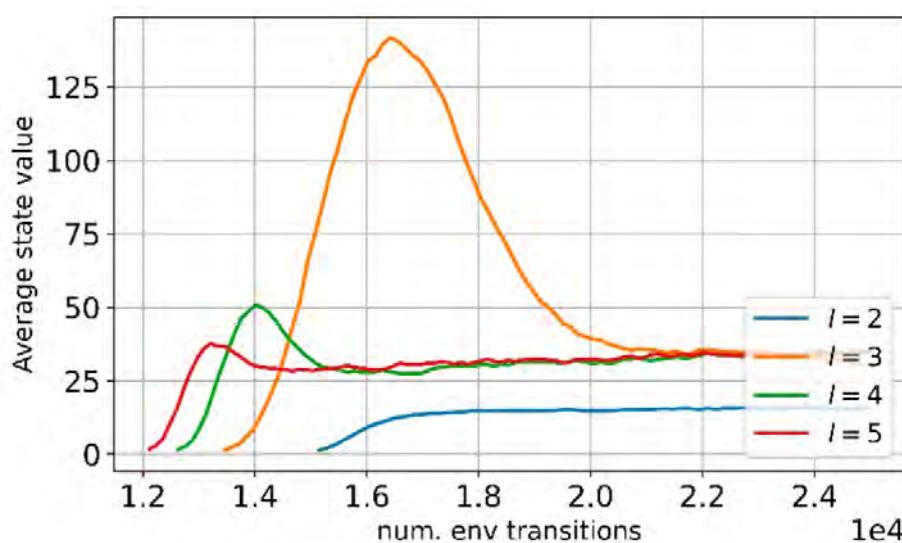


Рисунок 3.11 — Демонстрация распространения полезности из граничных состояний в алгоритме SAC с накоплением ошибок. Среда — это одномерный отрезок, по которому может перемещаться агент. Самое далёкое от начала состояние дает максимальное вознаграждение и достигается за 3 шага. Среднее значение функции полезности состояния резко возрастает, когда максимальное разрешенное количество шагов (l) равно 3.

В некоторых средах предлагаемый метод демонстрирует расхождение процесса обучения критика, что приводит к значительной переоценке полезностей и отсутствию сходимости самого процесса обучения стратегии. Данное явление связано с тем, что применяемая особая схемой генерации выборок из памяти используется для выбора переходов из модели среды. По этой схеме выбирается серия коротких траекторий из начальных состояний, определенных из наблюдений. Поскольку модель среды имеет неточности, последнее состояние каждой из этих коротких траекторий имеет значительную вероятность не оказаться в качестве начального состояния ни в одном из наблюдаемых переходов, выбранных из модели среды. Следовательно, критик не будет обновляться ни в одном из этих «граничных» состояний, которые будут накапливать ошибки в функции полезности при обновлениях критика в других точках. Высокие значения функции полезности на граничных состояниях будут распространяться в сторону наблюдаемых состояний, что приведет к катастрофическому завышению оценок. Данная гипотеза подтверждается на простой одномерной среде (см. рисунок 3.11), где видно, что этот эффект действительно имеет место.

Затем было проведено обучение и оценка модели NODE в задаче предсказания траектории с использованием среды *Reacher*. Было собрано 1000 траекторий длиной 50 шагов с использованием равномерной случайной стратегии, и на этом фиксированном наборе данных была обучена модель среды (см. рисунок 3.12а). После 500 итераций ошибка прогнозирования углов сочленений, усредненная на горизонте прогнозирования в 4 шага, падает ниже 1 градуса. Затем предлагаемый метод MBRL было оценен в среде *Reacher* с

использованием той же конфигурации динамической модели (см. рисунок 3.12b). Результаты демонстрируют, что предлагаемый метод одновременного обучения и планирования превосходит алгоритм SAC по эффективности выборок из среды, как и ожидалось, при этом успешно решая задачу. Базовый алгоритм обучения на основе модели Dreamer-v3 показывает меньшую производительность на среде *Reacher*.

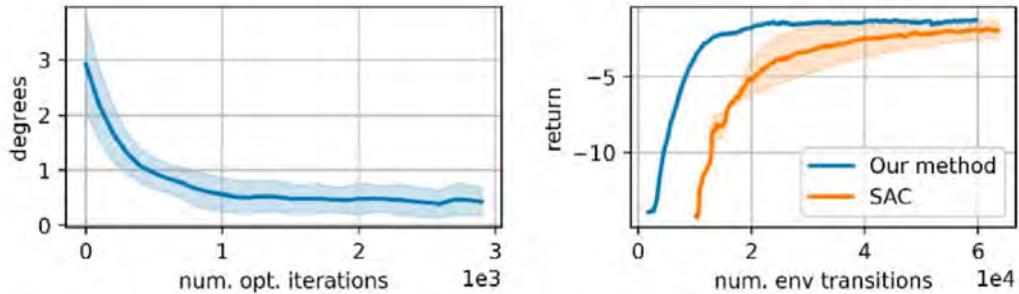


Рисунок 3.12 — Оценка метода в среде *Reacher*. (а) Средняя ошибка предсказания углов сочленений падает ниже 1 градуса на горизонте предсказания в 4 шага. (б) Предлагаемый метод обеспечивает более высокую эффективность выборок среды *Reacher*, требуя в 3 раза меньше элементов выборки, чем SAC.

Также было проведено более подробное сравнение предлагаемого подхода метода с алгоритмом Dreamer-v3 [421] в среде *Cheetah* (см. рисунок 3.13). Для всех сравниваемых подходов было предоставлены одинаковые векторные наблюдения. В рассматриваемой среде предлагаемый метод обучается быстрее при бюджете в 10^6 элементов выборки из среды (шагов). Однако при наличии дополнительных шагов Dreamer-v3 превосходит рассматриваемый метод. Также была проведена оценка предложенного метода на задаче открытия двери (см. рисунок 3.14) в среде *DoorOpen*. В наблюдение агента входит положение рабочего органа, а также положение и ориентация ручки двери. Рассматриваемый метод демонстрирует более высокое качество, чем Dreamer-v3. Алгоритму Dreamer-v3 не удалось решить задачу открытия двери с гиперпараметрами, предлагаемыми авторами по умолчанию.

В сложной среде *DoorOpen* рассматриваемый метод превосходит Dreamer-v3 и решает поставленную задачу. В среде *Cheetah* предложенный метод уступает по производительности Dreamer-v3, но, тем не менее, успешно обучается оптимальной последовательности действий.

В данном разделе диссертационного исследования представлен метод оптимизации стратегии на основе модели, который успешно решает задачу открытия двери. Основные отличия от предыдущих работ включают в себя расписание обучения и использование нейросетевых обыкновенных дифференциальных уравнений для моделирования динамики процессов в непрерывном времени. Проведены оценка работоспособности предлагаемого метода на различных задачах управления роботами и сравнение его с алгоритмом на основе модели Dreamer-v3 и алгоритмом без использования модели SAC. Предложенный метод превосходит эти методы в среде с задачей открытия двери и показывает сопоставимые результаты при решении других задач.

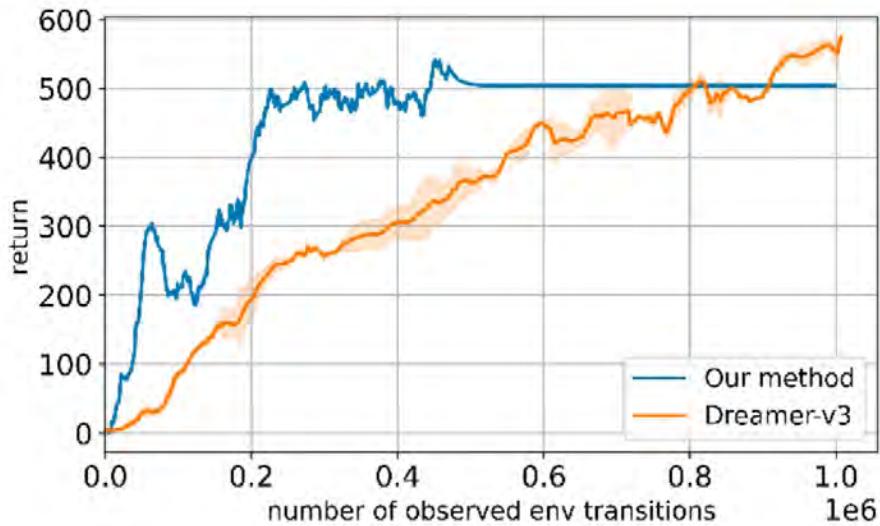


Рисунок 3.13 — В среде *Cheetah-run* рассматриваемый метод демонстрирует более высокую эффективность выборок из среды. Однако, при наличии 2×10^6 шагов Dreamer-v3 показывает более высокое качество по сравнению с предлагаемым методом. Все сравниваемые подходы используют идентичные векторные наблюдения.

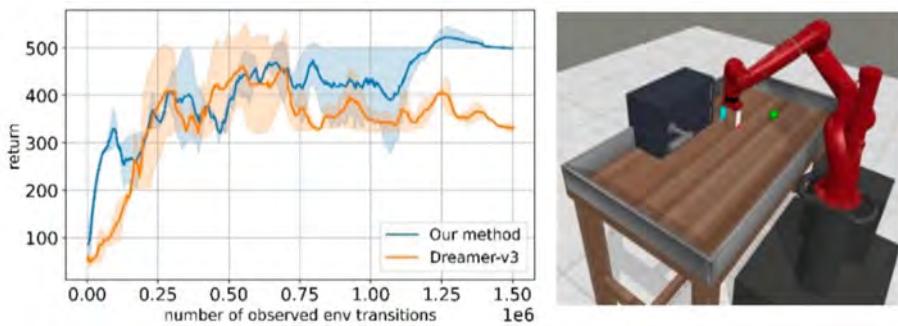


Рисунок 3.14 — (слева) Предлагаемый метод демонстрирует более высокое качество в среде *DoorOpen* по сравнению с Dreamer-v3 с размером модели XL (xlarge) и гиперпараметрами по умолчанию. Примечательно, что алгоритм Dreamer-v3 не решает задачу открытия двери.
(справа) Кадр из среды *DoorOpen*.

3.3 Реализация модели в многоагентных задачах

В данном разделе диссертационного исследования рассмотрена реализация интеграции обучения и планирования в архитектуре NSLP без использования иерархии навыков, но с поддержкой многоагентного режима (см. рисунок 3.15). Исследованы особенности построения и использования объектной модели мира M_O при обучении актора и критика с адаптацией подхода поиска по дереву Монте-Карло. В разделе предполагается, что в архитектуре NSLP уже проведена обработка первичной сенсорной информации o_t и агенту доступно графовое представление объектов (препятствий и других агентов). Основные результаты, изложенные в данном разделе, представлены в публикациях [2; 7; 12].

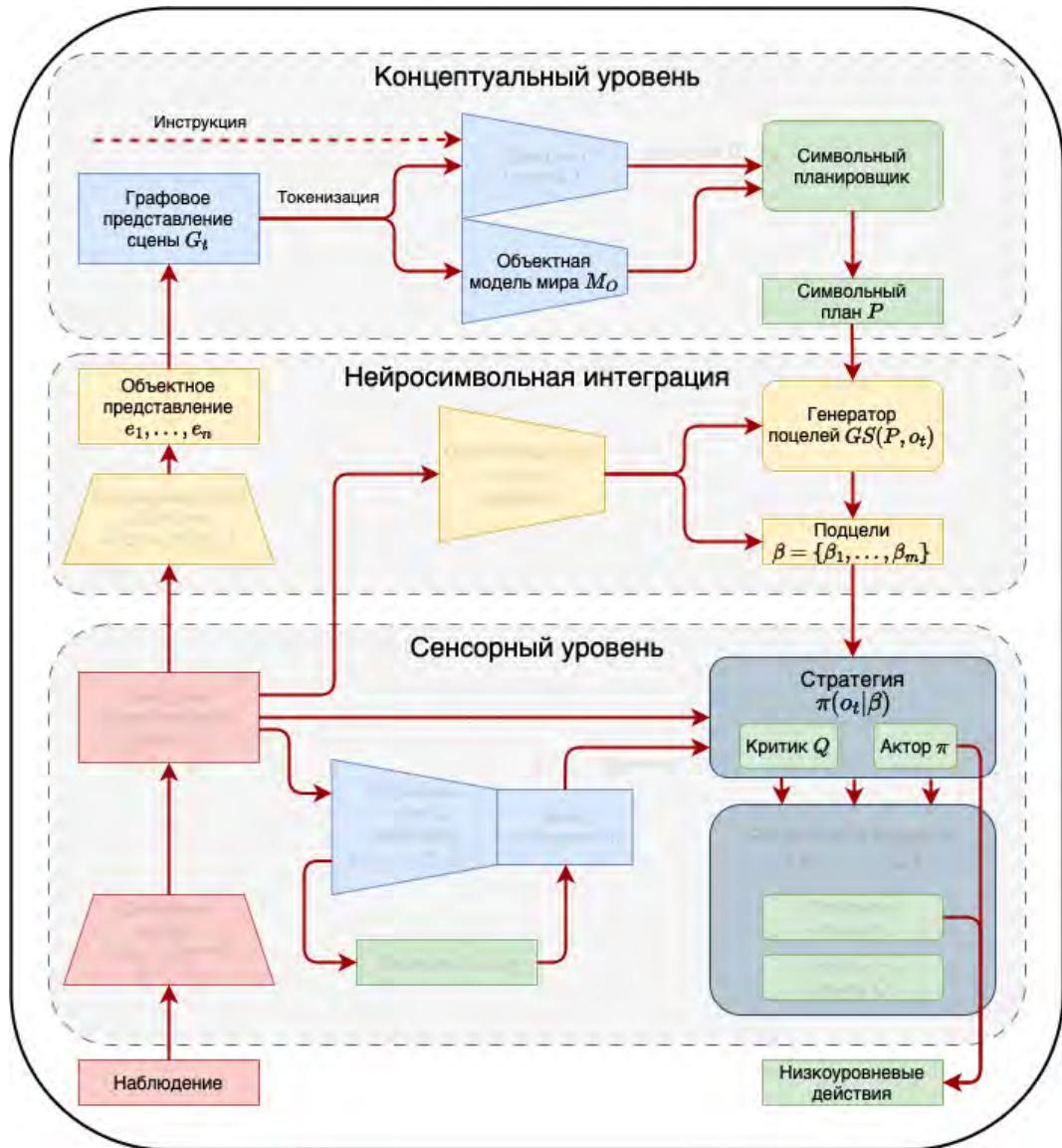


Рисунок 3.15 — Архитектура NSLP с выделенными блоками интеграции обучения стратегии и планирования по объектной модели мира M_O .

3.3.1 Поиск по дереву с эвристиками в задаче построения пути

Задача поиска пути естественным образом возникает в различных приложениях, таких как планирование движения автономных транспортных средств, сервисных робототехнических платформ [118] и т.д. Более того, в различных прикладных проблемах автоматизации производства желательно одновременное управление группой роботов. Таким образом, возникает задача многоагентного поиска пути (MAPF) [451], для которой известны многочисленные подходы, как обучаемые, так и необучаемые (см. обзор в [70]).

В данном разделе предполагается исследовать, как механизм поиска по дереву Монте-Карло (MCTS) [112] может быть применен к задаче MAPF в контексте архитектуры NSLP. MCTS можно рассматривать как гибридный подход, объединяющий планирование и обучение и уже успешно применяющийся для решения различных задач с известной

комбинаторной структурой, часто превосходя по эффективности самых современных конкурентов (см. раздел 1.3). Если MCTS показывает высокую производительность в комбинаторных задачах, это открывает двери для внедрения различных методов глубокого обучения, которые могут еще больше повысить его эффективность (аналогично тому, как глубокое обучение использовалось для повышения качества MCTS при игре в настольные игры, такие как го, достигая производительности на уровне и выше человека-эксперта [423]). В данном разделе будет рассмотрено то, как MCTS может быть адаптирован к MAPF в случае его использования как объектной модели мира в архитектуре NSLP и насколько хорошо он может работать по сравнению со стандартными подходами, основанными на эвристическом поиске.

С этой целью представлен оригинальный вариант MCTS, в котором используются два ключевых компонента. Первый компонент — это специальные механизмы формирования вознаграждения и его распространения, адаптированные к постановке задачи MAPF. Второй компонент — механизм того, как уменьшается коэффициент ветвления дерева поиска путем разложения совместных действий всех агентов на последовательную комбинацию действий отдельного агента. Многочисленные эксперименты подтверждают, что полученный алгоритм способен успешно решать сложные задачи MAPF и превосходит базовый алгоритм на основе классического поискового решателя.

Поиск пути и алгоритм MCTS

Многоагентный поиск путей в его классическом варианте [451] предполагает, что множество K агентов перемещается по графу занятости $G = (V, E)$, где различаются n положений агента, сопоставленных с вершинами графа. Время дискретизировано, и на каждом временном шаге агент может ждать в своей текущей вершине или перейти в соседнюю. Последовательность таких действий называется планом. Два плана для разных агентов бесконфликтны, если следующие за ними агенты никогда не меняют вершины местами на одном и том же временном шаге и никогда не занимают одну и ту же вершину на одном и том же временном шаге (т.е. не допускаются коллизии типа *ребро* и типа *вершина*). Задача состоит в том, чтобы найти набор планов $Plans = \{plan_1, plan_2, \dots, plan_n\}$, по одному для каждого агента, в которых все агенты достигают своих целей и каждая пара планов бесконфликтна. Как правило, в MAPF требуется минимизировать одну из следующих метрик: $SOC = \sum_{i=1}^n cost(plan_i)$ или $makespan = \max_i cost(plan_i)$, где $cost(plan_i)$ — стоимость индивидуального плана, т.е. количество временных шагов, которое потребовалось агенту i для достижения своей цели.

В данном разделе рассматривается формулировка задачи MAPF как задачи последовательного принятия решений, когда на каждом временном шаге определяются действия для агентов (на основе некоторой стратегии выбора действий), затем эти действия выполняются, после чего цикл повторяется до тех пор, пока все агенты не достигнут своих

целей или достигается некоторый предопределенный порог по количеству временных шагов. Такая постановка задачи обычно формализуется с помощью многоагентного марковского процесса принятия решений (MAMDP) [459], который будет описан далее.

Определение 3.3.1 (Многоагентный марковский процесс принятия решений [459]). *Многоагентный марковский процесс принятия решений (MAMDP) — это кортеж из 6 элементов $\langle S, U, T, r, n, \gamma \rangle$, где n — количество агентов, а $\mathbf{U} = U_1 \times U_2 \times \dots \times U_n$ — это совместное действие, которое формируется путем объединения индивидуальных действий всех агентов, S — это множество состояний среды, $T(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \rightarrow [0, 1]$ — это функция переходов в среде, $r : S \times \mathbf{U} \rightarrow \mathbb{R}$ — это функция вознаграждения, разделяемая между всеми агентами, $\gamma \in [0, 1]$ — коэффициент дисконтирования.*

В рассматриваемой постановке задачи предполагается, что каждое состояние $s \in S$ содержит информацию о местоположении всех агентов, то есть агенты имеют доступ к графовому представлению сцены G_t . Функция переходов определяет вероятность перехода в состояние s' , когда совместное действие $\mathbf{u} \in \mathbf{U}$ выполняется в состоянии s . Функция вознаграждения определяет, какое вознаграждение (скалярное значение) получат агенты, если они выполнят определенное действие \mathbf{u} в определенном состоянии среды s .

Задача состоит в том, чтобы создать стратегию π , которая определяет, какие действия агент должен предпринять в различных состояниях среды, чтобы максимизировать ожидаемую отдачу G за эпизод длиной K , как определено ниже:

$$G_t = \sum_{k=0}^{K-t-1} \gamma^k r_{t+k+1} \text{ при условии } s_t \in S, \quad (3.1)$$

где r_{t+k+1} — вознаграждение, полученное на временном шаге $t + k + 1$, а s_t представляет состояние окружающей среды на временном шаге t . Как правило, стратегия является стохастической, т.е. она может сопоставлять состояния с распределением действий: $\pi : S \times \mathbf{U} \rightarrow [0, 1]$. Учитывая стратегию каждого агента, точное (совместное) действие выбирается из распределения на каждом временном шаге.

Могут быть предложены различные подходы к формированию стратегии π для задачи MAPF. Например, можно вызывать полный/оптимальный решатель MAPF на каждом временном шаге и выбирать первое совместное действие из решения MAPF. Другим подходом может быть вызов n эгоцентричных поисков пути для одного агента и построение совместного действия путем объединения первых действий, составляющих путь n для одного агента, которые не учитывают других агентов. В данном разделе диссертационного исследования будет использоваться такой подход в качестве базового алгоритма для сравнения. Еще одним направлением, на которое следует обратить внимание, является использование обучения с подкреплением без модели среды для формирования стратегии. Это, однако, может оказаться сложным в условиях многоагентной постановки задачи, и потребуются нетривиальные модификации алгоритмов, основанных на обучении [445]. В данной работе будет предложен иной подход, заключающийся в адаптации поиска по дереву Монте-Карло к задаче многоагентного поиска путей, обладающий высокой эффективностью поиска, особенно в случае комбинаторных задачах с высоким коэффициентом ветвления.

Поиск по дереву Монте-Карло (MCTS) — это известная поисковая парадигма, которая хорошо подходит для задач последовательного принятия решений. В сочетании с самыми современными методами машинного обучения алгоритмы на основе MCTS недавно продемонстрировали производительность в различных настольных играх и видеоиграх на уровень выше человека (см., например, [418; 423]). Поскольку MCTS опирается на понятие вознаграждения, его также можно отнести к методу обучения с подкреплением, основанному на модели (как это было показано в разделе 1.3), который использует вознаграждение (вознаграждения) для обучения выбору наиболее перспективных действий в вершине дерева.

В общем виде алгоритм MCTS выбирает действие с учетом состояния окружающей среды, основываясь на симуляционном моделировании того, как изменится окружающая среда и какие вознаграждения будут получены при последовательном выполнении различных (случайных) действий. При большом пространстве действий невозможно смоделировать все возможные последовательности действий за ограниченное время. С этой целью MCTS строит дерево (ограниченной ширины и глубины), содержащее наиболее перспективные варианты действий, которые необходимо выполнить в среде (частичные планы). Узлы в этом дереве соответствуют состояниям среды, а ребра — действиям. Корнем дерева является текущее состояние, т.е. то состояние среды, для которого агенту нужно выбрать действие в данный момент времени.

Собственно алгоритм MCTS состоит из четырех шагов, которые выполняются итеративно и предназначены для одновременного построения и прохождения по дереву поиска: выбор, расширение, симуляция и обратное распространение.

Шаг выбора представляет собой нисходящее движение по построенному на данный момент дереву поиска. Концептуально это можно рассматривать как процесс выбора наиболее перспективного частичного плана для рассмотрения. Чтобы сбалансировать исследование среды (т.е. выбор частей дерева поиска, которые ранее не рассматривались) и использование накопленного опыта (т.е. выбор частичных планов, которые характеризуются наибольшими вознаграждениями), MCTS полагается на оценку полезности узлов с использованием метода верхних доверительных границ (которые первоначально был предложен в контексте задачи о многоруких бандитах [140]). В частности, обычно используется верхняя доверительная граница для деревьев поиска (UCT) [359].

Шаг расширения. Когда по дереву поиска шаги выбора достигли конечного узла, последний расширяется путем выбора действия, которое ранее не было опробовано алгоритмом, и добавления нового узла к дереву.

Шаг симуляции. Добавленный узел оценивается путем симуляции действий с использованием случайной стратегии.

Шаг обратного распространения. Полученное в результате симуляции итоговое вознаграждение (или отдача) по равномерному правилу распространяется обратно по дереву.

Итеративный процесс, чередующий все четыре шага, повторяется до тех пор, пока не будет исчерпан бюджет времени. Когда это происходит, для выполнения в среде выбирается

действие, соответствующее наиболее посещаемому исходящему ребру корневого узла. Более подробная информация об алгоритме MCTS доступна изложена в обзорной статье Брауна и др. [112]. В данном разделе диссертационного исследования представлена оригинальная адаптация алгоритма MCTS для многоагентного поиска пути.

Адаптация алгоритма MCTS к задаче многоагентного поиска пути

Рассмотрим n однородных агентов, перемещающихся в среде, которая представлена в виде 4-связной клеточной карты (графа), состоящей из свободных и заблокированных клеток (последние соответствуют статическим препятствиям среды). Временная шкала дискретизирована, и на каждом временном шаге агент может ждать в текущей клетке или перейти в одну из соседних клеток, если она не заблокирована. Когда два агента хотят переместиться в одну и ту же свободную клетку, только один из них (случайный) выполняет данное действие, в то время как другой остается там, где был. Таким образом, переходы состояний являются стохастическими.

Первоначально агенты располагаются в уникальных начальных клетках, и для каждого агента указывается уникальная целевая клетка. Когда агент достигает цели, он удаляется из среды. Это предположение является одной из вариаций в задаче MAPF и служит правдоподобным условием для различных практических применений. Примером могут служить роботы на складе, которым необходимо добраться до зарядных станций. Как правило, эти зарядные станции расположены по периметру рабочей зоны, поэтому можно считать, что робот, который достигает зарядной станции, покидает рабочее пространство.

Как обычно, задача состоит в том, чтобы сформировать стратегию, которая будет сопоставлять состояния среды с совместными действиями агентов. Здесь состояние среды — это наблюдаемая агентом клеточная карта плюс позиции всех агентов на ней. Совместное действие — это комбинация индивидуальных действий отдельных агентов. Более того, предполагается, что задано ограничение в K_{max} временных шагов, и эпизод заканчивается, когда проходит это количество временных шагов (независимо от того, где к этому времени находятся агенты).

Чтобы оценить, насколько хорошо стратегия справляется с поставленной задачей, были использованы следующие показатели, которые вычисляются в конце эпизода:

Коэффициент успешной кооперации (CSR) — это бинарный показатель успешности, т.е. он может быть равен 0 или 1, что указывает, удается ли всем агентам достичь своих целей;

Индивидуальный коэффициент успеха (ISR) — это доля агентов, которым удалось достичь своей цели до окончания эпизода.

Длина эпизода (EL) — это количество шагов, предпринятых агентами для выполнения задачи. Если не все агенты достигают своих целей, EL присваивается значение K_{max} .

При сравнении различных стратегий лучшей считается та, которая обеспечивает более высокое значение CSR/ISR и более низкое значение EL. Необходимо обратить внимание, что в данной работе не ставится цель получить оптимальную стратегию, — необходимо лишь найти лучшую субоптимальную стратегию поведения агента в такой многоагентной среде.

Простая адаптация алгоритма MCTS к рассматриваемой задаче MAPF заключается в следующем. Узлы в дереве представляют состояния, которые являются позициями агентов на клеточной карте, ребра соответствуют совместным действиям. Таким образом, коэффициент ветвления дерева равен $|A|^n$, где $|A|$ — количество возможных индивидуальных действий, а n — количество агентов. Для большого числа агентов это приводит к большим вычислительным затратам. С этой целью было предложено разложить совместное действие на отдельные элементы в дереве, как описано далее.

Следующая проблема, которая возникает, когда необходимо адаптировать MCTS для задачи MAPF, заключается в том, как вычислить вознаграждение в конце фазы симуляции, т.е. после фазы, когда агенты совершают случайные действия. Как правило, MCTS применяется к антагонистическим играм, где результатом эпизода является либо победа (вознаграждение равно 1), либо проигрыш (вознаграждение равно 0), либо ничья (вознаграждение равно 1/2). Таким образом, в рассматриваемом случае можно сконструировать следующую функцию вознаграждения: $R = n_{finished}/n$, где $n_{finished}$ — количество агентов, достигших своих целей. Однако такое вознаграждение крайне разреженно, т.е. при многочисленных запусках симулятора вознаграждение будет равным нулю или очень близким к нулю (поскольку очень сложно достичь цели, случайным образом перемещаясь по клеточной карте за ограниченное количество шагов). Таким образом, будет очень сложно направить поиск на наиболее перспективные участки дерева поиска. С этой целью было введено дополнительное внутреннее вознаграждение для каждого агента, основанное на том, как часто он достигает клеток, лежащих на кратчайшем (индивидуальном) пути к своей цели. Такое формирование вознаграждения вынуждает агентов демонстрировать «жадное» поведение, направленное на достижение цели, оставляя, однако, возможность агенту отклониться от «жадного» пути, когда это необходимо, например, когда некоторым агентам нужно уступить место другим агентам.

Алгоритм MCTS эффективен в антагонистических играх, потому что в них обычно имеется ограниченное количество ходов и определенный исход (проигрыш, победа, ничья). Однако в задаче MAPF эпизоды могут длиться дольше, и часто случайно перемещающиеся агенты не достигают своих целей. В таких симуляциях агенты могут не получать положительный сигнал вознаграждения, который направляет поиск. Как было упомянуто выше, был предложен механизм, который решает эту проблему, поощряя достижение подцели с помощью внутренних вознаграждений, используя классический алгоритм планирования, такой как A*. Алгоритм планирования используется для поиска пути, игнорируя других агентов. Выбранная подцель размещается на небольшом расстоянии от агента, например, в двух шагах от текущего положения.

Итоговая функция вознаграждения выглядит следующим образом:

$$r_t = \sum_{i=0}^{n-1} \frac{r_t^{a_i}}{n}, \quad r_t^{a_i} = \begin{cases} r_{target}, & \text{если агент } a_i \text{ достиг цели,} \\ r_{subgoal}, & \text{если агент } a_i \text{ достиг подцели,} \\ 0, & \text{иначе.} \end{cases} \quad (3.2)$$

Достижение подцели приводит к небольшому положительному вознаграждению (обозначаемому $r^s \in R$), которое меньше глобального целевого вознаграждения (обозначаемого $r^g \in R$), так что $r^s \ll r^g$. Вознаграждение поступает из среды для всех агентов при выполнении совместного действия. Это вознаграждение делится на количество агентов в среде, чтобы обеспечить соответствие бонусу за исследование UCT. Если агент слишком далеко отходит от подцели, позиция подцели пересчитывается.

Наивное применение алгоритма MCTS к задаче MAPF предполагает генерацию $|A|^n$ дочерних узлов при расширении пространства поиска в дереве, где $|A|$ — количество отдельных действий, а n — количество агентов. При решении конкретных задач это оказывается очень неэффективным. Чтобы уменьшить коэффициент ветвления, предлагается подход к декомпозиции, представленный на рисунке 3.16 (необходимо отметить, что это напоминает метод, предложенный в [592]). Для каждого агента варианты выбора действий рассматриваются как отдельные узлы в дереве. Действия, накопленные таким образом, применяются в симуляционной среде в тот момент, когда последний агент выбирает свое действие (эти узлы представлены на рисунке в виде квадратов). Таким образом, состояние среды определяется текущими позициями агентов и накопленными действиями других агентов. Если накопленных действий недостаточно для выполнения совместного действия для запуска симуляции, оставшееся действие выбирается в соответствии со стратегией по умолчанию. Кроме того, в дереве ограничивается выбор действий, которые приводят в статические препятствия на клеточной карте.

Выбор узла выполняется с использованием бонуса за исследование в соответствии с оценкой верхней границы UCB:

$$r^{USB} = \frac{V_j}{k_j} + C_p \sqrt{\frac{2 * \ln(k)}{k_j}}, \quad (3.3)$$

как в классических MCTS, но теперь для каждого отдельного узла. В этом уравнении V_j — общая накопленная полезность j -го дочернего узла, C_p — коэффициент исследования среды, k — количество посещений родительского узла, а k_j — количество посещений дочернего узла.

Последовательный подход к выбору действия также изменяет расчет отдачи G (см. уравнение 3.1), где теперь дисконтирование происходит только один раз между выполнением совместных действий в среде (между квадратными вершинами на рисунке 3.16). Статистика в вершинах обновляется с использованием того же значения функции вознаграждения после накопления общего действия. Возвращаемое значение G , полученное во время симуляции с использованием стратегии по умолчанию (случайной), вычисляется стандартным способом. Функция полезности в стиле обучения с подкреплением (т.е. с дисконтированием $\gamma < 1$) используется для определения приоритета более коротких путей.

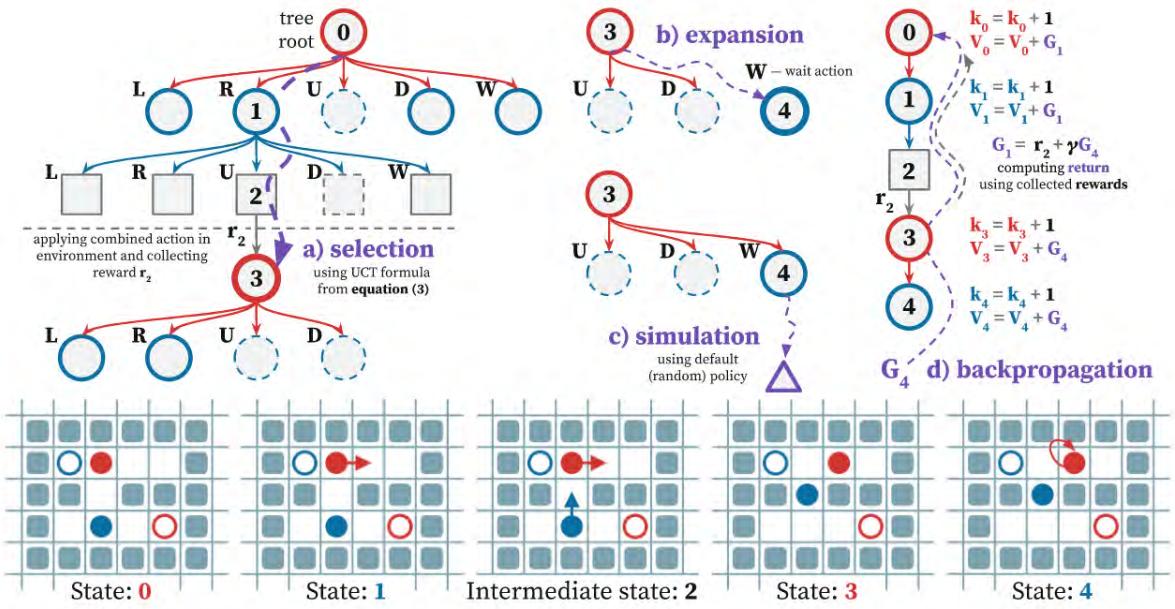


Рисунок 3.16 — Подход МАМCTS адаптирует алгоритм MCTS для многоагентных задач, рассматривая варианты выбора действий для каждого агента как отдельные узлы в дереве, тем самым уменьшая коэффициент ветвления. На рисунке показаны все четыре этапа MCTS: а) выбор (selection); б) расширение (expansion); в) симуляция (simulation); и д) обратное распространение (backpropagation) для многоагентной постановки задачи. Действия, ведущие к вершинам, показанным пунктирными линиями, не рассматриваются на этапе выбора. Заполненные круги представляют агентов, в то время как пустые круги представляют соответствующие им цели.

Экспериментальное исследование алгоритма МАМCTS

Для проведения экспериментального исследования представленного алгоритма МАМCTS была использована среда POGEMA¹ [67; 501], многоагентная среда поиска путей с открытым исходным кодом. Изначально разработанная для задач частичной наблюдаемости, она хорошо подходит для используемой постановки с полной наблюдаемостью. Вместо выделения состояний для последующего восстановления была реализована специальная функция отката, которая восстанавливалась среду до ее исходного состояния, что необходимо для древовидного планирования на этапе симуляции. Этот подход обеспечивает лучшую вычислительную производительность, чем простое выделение состояний.

Для оценки эффективности стратегии агента использовались карты двух типов. Первый класс карт включает клеточные карты со случайно заблокированными клетками. Хотя случайные карты хороши своим разнообразием, они не создают сложных проблем для алгоритма, поскольку узкие проходы, требующие согласованного поведения агентов, редко появляются на карте. Чтобы продемонстрировать потенциал кооперативного поведения, были использованы процедуры, предложенные в [7], и создан набор сложных карт,

¹<https://github.com/AIRI-Institute/pogema>

отсортированных по количеству пересечений индивидуальных путей 16 агентов. При генерации карт использовалось 2000 случайных инициализаций среды, и из них выбрано 100 самых сложных с размерами 16×16 . Обычно на этих картах есть только один центральный проход, что вынуждает агентов действовать согласованно. На рисунке 3.17(a) представлен пример такой кооперативной карты. Для их создания использовалась функция, представленная в среде POGEMA с плотностью препятствий 0,3. Была также проведена постобработка этого набора с заполнением пустых компонент препятствиями размером меньше пяти клеток.

В качестве второго набора тестовых карты использовались карты лабиринтов, сгенерированные с помощью пакета LABMAZE², который содержит общие схемы лабиринтов и комнат с несколькими входами. Эти карты включают множество узких проходов и по дизайну являются более сложными, чем случайные карты. В этом случае специально не проводился отбор сложных исходных данных для этого набора. Предполагается, что сложность этих карт примерно такая же, как и у кооперативных случайных карт. При создании карт для второго набора использовался размер 15×15 (поскольку пакет LABMAZE поддерживает только нечетные размеры). Пример сгенерированной карты представлен на рисунке 3.17 (b). Параметры, используемые для создания набора карт labmaze, были следующие: max_rooms: 30, has_doors: True, room_min_size: 5, room_max_size: 5, simplify: False, min_component_size: 4, retry_count: 1000, extra_connection_probability: 0.0.

Было проведено сравнение четырех алгоритмов: централизованный MCTS, MAMCTS, MAMCTS с подцелями и модифицированная A* [342]. Централизованный MCTS — это наивный вариант MCTS, где выбор действия в дереве происходит в пространстве совместных действий всех агентов. Второй подход — MAMCTS, где пространство действий сокращено и на каждом шаге расширения узла дерева рассматривается только один агент. Третий вариант — MAMCTS с подцелями, в котором используется модифицированная функция вознаграждения, включающая сигнал для достижения подцелей. Алгоритм A* первоначально был разработан для эвристического планирования пути в задачах с одним агентом. Алгоритм строит план на полностью наблюдаемой карте, рассматривая других агентов как препятствия. Первое действие из плана алгоритма выбирается в качестве действия в окружающей среде. Предварительные эксперименты показали, что этот простой подход хорошо работает для задач с небольшим количеством агентов и картами без узких проходов. Однако, если агентам необходимо учитывать действия других агентов, например, один агент должен пропустить другого, этот наивный алгоритм сталкивается со значительными проблемами. Чтобы преодолеть эти проблемы, в нем была введена дополнительная эвристика — если последние два действия алгоритма не приводят к цели, агент выбирает случайное действие вместо этого.

Во время тестирования все подходы, основанные на MCTS, получали вознаграждение в размере $r_{target} = +1$ за каждого агента, достигшего цели (очевидно, что подход A* не использует эту информацию). Эпизод взаимодействия с окружающей средой длился 64 шага. В качестве системы разрешения конфликтов использовался подход со случайнym

²<https://github.com/deepmind/labmaze>

приоритетом. Если несколько агентов пытались переместиться в одну и ту же клетку на одном и том же временном шаге, один агент выбирался случайным образом, чтобы занять клетку, в то время как остальные оставались на своих исходных позициях. При построении дерева было запрещено выбирать действия, которые приводят в клетку со статическим препятствием. Во время выполнения алгоритмы MCTS выполняли 1000 итераций шагов выбор-расширение, используя коэффициент исследования $C_p = 1$. При формировании статистики по полезности был использован коэффициент дисконтирования $\gamma = 0,9$.

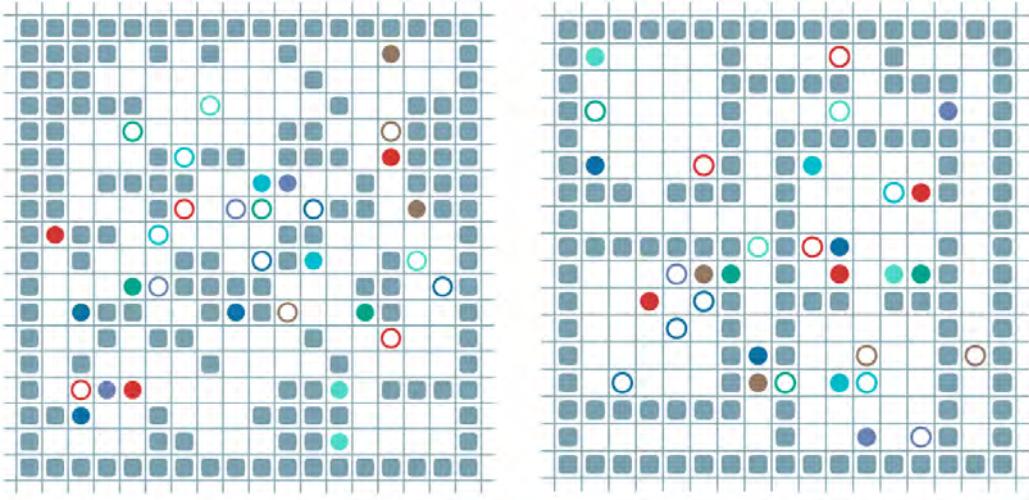


Рисунок 3.17 — Примеры карт, использованных во время обучения: (а) Кооперативные случайные карты, которые были выбраны в соответствии с их сложностью. (б) Карты лабиринтов, сгенерированные с помощью пакета labmaze. Агенты представлены заполненными кругами, а их цели — пустыми кругами. У каждого агента есть своя цель.

Алгоритм МАМCTS с подцелями выдает сигнал вознаграждения в размере $r = +0,1$ за достижение подцели, которая представляет собой клетку, расположенную в двух шагах от агента по кратчайшему пути (рассматривая клетки, занятые другими агентами, как проходимые). Подцель пересчитывается каждый раз, когда агент достигает ее или удаляется от нее на 2 шага. Данный горизонт планирования был определен опытным путем в предварительном эксперименте. Также использовалось ограничение в 10 шагов для этапа симуляции для этого алгоритма, что значительно ускоряет его без ущерба для итогового результатов. Для других алгоритмов это ограничение не применялось, поскольку оно значительно ухудшало их результаты.

Запуск алгоритмов на кооперативных случайных картах приводит к результатам, представленным в таблице 4. Здесь для каждого алгоритма приведены усредненные значения всех описанных выше показателей. Во-первых, необходимо отметить, что МАМCTS превосходит централизованный MCTS. Пространство действий централизованного алгоритма MCTS слишком велико, поскольку экспоненциально зависит от количества агентов в среде. Планирование всего на один шаг вперед в дереве требует вычисления результатов для 5^n возможных узлов, что значительно замедляет процесс поиска. Разница особенно заметна на картах с большим количеством агентов. Сигнал вознаграждения, когда агент достигает итого узла, также довольно редок, что можно увидеть, сравнив результаты

алгоритмов MAMCTS и MAMCTS с подцелями. Дополнительный сигнал вознаграждения решает проблему долгосрочного планирования.

Таблица 4 — Производительность различных алгоритмов сравнивалась на 100 совместных случайных картах размером 16×16 .

Agents	JOINT MCTS			MAMCTS			SUBGOAL MAMCTS			A*		
	ISR	CSR	Ep. len.	ISR	CSR	Ep. len.	ISR	CSR	Ep. len.	ISR	CSR	Ep. len.
4	0.31	0.04	49.37	0.43	0.0	41.31	1.0	1.0	18.8	0.84	0.75	24.73
8	0.29	0.0	50.83	0.48	0.0	40.08	0.99	0.93	21.7	0.66	0.39	32.3
16	0.24	0.0	53.3	0.4	0.0	44.47	0.9	0.21	30.14	0.4	0.0	44.21

Модификация алгоритма A* работает хорошо и превосходит все алгоритмы поиска по дереву за исключением MAMCTS с подцелями. Разница особенно заметна на картах с количеством агентов 4 и 8. Конфликты в таких случаях возникают редко, и этот алгоритм достаточно эффективен даже при случайных действиях для их разрешения.

Сравнение алгоритмов на картах лабиринтов в значительной степени повторяет результаты предыдущего эксперимента. Результаты представлены в табл. 5. Подход MAMCTS с подцелями остается лидером среди рассмотренных алгоритмов, за ним следует модифицированный алгоритм A*. Также примечательно, что этот набор карт оказался немного проще, чем кооперативные случайные карты.

Кроме того, было проведено сравнение вычислительной эффективности всех алгоритмов. т.е. было вычислено, сколько времени потребовалось каждому алгоритму, чтобы выбрать действия для 16 агентов (в среднем по всем типам карт). В среднем, централизованному MCTS требуется 12,1 секунды для принятия решения о действиях, MAMCTS занимает 12,7 секунды, а MAMCTS с подцелями — 4,2 секунды из-за ограниченного количества шагов в симуляции. Как и ожидалось, алгоритм A* работает лучше всего: на выбор действий уходит всего 0,02 секунды.

Таблица 5 — Результаты тестирования алгоритмов на 100 лабораторных картах лабиринтов размером 15×15 . Алгоритм подцели MAMCTS показал лучшие результаты.

Agents	JOINT MCTS			MAMCTS			SUBGOAL MAMCTS			A*		
	ISR	CSR	Ep. len.	ISR	CSR	Ep. len.	ISR	CSR	Ep. len.	ISR	CSR	Ep. len.
4	0.39	0.0	48.25	0.54	0.0	37.74	1.0	1.0	17.17	0.9	0.79	22.41
8	0.32	0.0	50.82	0.55	0.0	39.16	0.98	0.89	20.18	0.83	0.54	26.42
16	0.22	0.0	54.19	0.5	0.0	42.02	0.94	0.46	25.64	0.58	0.07	38.62

В заключение можно сделать вывод, что предложенный алгоритм MAMCTS с подцелями, показал лучшие результаты по сравнению со всеми другими алгоритмами. Снижение коэффициента ветвления за счет разделения пространства действий между агентами, а также добавления дополнительного сигнала вознаграждения, который направляет агента к цели, были ключевыми факторами успеха этого алгоритма. С другой

стороны, текущая реализация MCTS с подцелями, очевидно, медленнее по сравнению с базовым алгоритмом, основанным на поиске.

В данном разделе алгоритм поиска по дереву Монте-Карло был применен к задаче многоагентного поиска пути и были предложены методы для повышения его производительности. Разработанный алгоритм MAMCTS с подцелями, реализующий модуль одновременного обучения и планирования в архитектуре NSLP в многоагентной постановке, превосходит стандартные реализации MCTS и простые алгоритмы планирования на различных вариантах карт, особенно в сценариях с высокой плотностью агентов. Несмотря на то что MAMCTS может быть не таким быстрым, как отдельные стратегии поиска путей, основанные на алгоритме A*, он оказывается эффективным при наличии достаточных вычислительных ресурсов и бюджета времени. В дальнейшем это направление диссертационного исследования можно развивать в направлении внедрении нейросетевых аппроксиматоров для ускорения алгоритма путем оценки стратегии и функции полезности (см. следующий параграф). Кроме того, применение MCTS к многоагентной задаче открывает возможности для решения более сложных задач в средах с изменяющейся топологией карты и агентами, обладающими дополнительными действиями по изменению среды (манипуляции объектами).

3.3.2 Многоцелевой многоагентный поиск по дереву с планированием во внутренней среде

Задача многоцелевого многоагентного поиска пути

В продолжение рассмотрения вопроса реализации модуля одновременного обучения и планирования в NSLP в многоагентной постановке более подробно рассмотрим методы в многоцелевой (lifelong) постановке: обучаемые методы MAPF и использование MCTS для многоагентных систем, в частности, в задаче MAPF.

В данном параграфе рассматривается многоцелевой вариант задачи MAPF (LMAPF), где сразу после того, как агент достигает своей цели, ему назначается новая цель (посредством процедуры внешнего назначения) и он должен продолжать движение к новой цели. Таким образом, LMAPF обычно запрашивает набор из K начальных планов и обновляет план каждого агента, когда тот достигает текущей цели и получает новую. В тех случаях, когда на каждом этапе достигается какая-то цель, планы необходимо обновлять постоянно (т.е. на каждом временном шаге). Так, для данного исследования задача MAPF рассматривается как задача последовательного принятия решений: на каждом временном шаге должно быть принято решение о следующем действии (для всех агентов). Предполагается, что модуль назначения целей является внешним по отношению к множеству

агентов и их поведение не влияет на назначение целей. Также предполагается, что любой экземпляр задачи LMAPF дополнительно характеризуется продолжительностью эпизода L , измеряемой во временных шагах. По истечении L временных шагов выполнение экземпляра считается завершенным (несмотря на то что некоторые агенты находятся на пути к поставленным в данный момент целям). Обычные метрики качества MAPF, такие как $SOC = \sum_{i=1}^n cost(plan_i)$ или $makespan = \max_i cost(plan_i)$ (где $plan_i$ — построенный план i -м агентом), напрямую неприменимы к LMAPF. Наиболее часто используемым показателем эффективности в LMAPF является пропускная способность (throughput), которая представляет собой среднее количество целей, достигаемых агентами за один шаг. Технически это вычисляется как отношение продолжительности эпизода к общему количеству достигнутых целей.

Среди недавних работ, посвященных MAPF, одной из первых, которые были специально посвящены созданию обучаемого решателя задачи MAPF, была работа [508]. Сочетание обучения с подкреплением и обучения на основе демонстраций, полученных от экспертов, было использовано для создания обучаемой стратегии под названием PRIMAL, адаптированной для решения обычных задач MAPF. Позже в работе [507] была представлена улучшенная версия этого решателя, PRIMAL2. Последний был оснащен специальными методами анализа коридоров, направленными на то, чтобы избежать тупиков в узких участках свободных областей карты, и поддерживал многоцелевую постановку задачи MAPF (поэтому далее PRIMAL2 был выбран в качестве одного из базовых алгоритмов, с которым было проведено сравнение предлагаемого метода). Среди других обучаемых решателей для задачи MAPF, которые используют обучение с подкреплением для получения стратегии принятия решений, можно назвать [287; 438]. Методы, основанные на обучении и представленные в работах [295; 414; 450], добавляют агентам коммуникационные возможности, т.е. позволяют агентам обмениваться дополнительными сообщениями для устранения взаимных блокировок и предотвращения бесконечных циклов обмена позициями. В данном разделе работы проводится сравнение с одним из самых последних методов, основанных на коммуникации, методом SCRIMP [556]. Однако необходимо отметить, что предлагаемый метод не полагается на коммуникацию агентов.

Как уже было сказано в предыдущем параграфе, первоначально алгоритмы поиска по дереву Монте-Карло (MCTS) продемонстрировали свою эффективность в соревновательных играх с полной информацией, таких как шахматы или Go [423]. Более поздние версии MCTS используют глубокие нейронные сети для аппроксимации полезности игровых состояний вместо того, чтобы полагаться исключительно на симуляции. Эти подходы также показали многообещающие результаты в сценариях с одним агентом, где агенты могут формировать модель окружающей среды и играть, например, в игры Atari [418; 420]. Помимо игр, методы MCTS нашли применение в других областях, таких как оптимизация алгоритма умножения матриц [228] и доказательство теорем с использованием подхода гипердерева [324]. Кроме того, методы MCTS продемонстрировали применимость и в робототехнике [210; 446].

Несмотря на растущий интерес к использованию алгоритма MCTS для многоагентных задач, его применение в задаче MAPF было ограниченным. Например, в работе [656]

авторы предлагают многоагентную реализацию MCTS для анонимного MAPF в клеточной среде. Используемая в этой работе среда имеет плотный сигнал вознаграждения (агент, достигший любой цели на карте, получал вознаграждение и эпизод завершился). В ней нет препятствий, что облегчает предотвращение столкновений. Авторы строят отдельное дерево для каждого агента, используя классический алгоритм. Затем они совместно выбирают наилучшие действия (формируют план) из деревьев в симуляции, чтобы получить истинные оценки решения и обновить деревья с учетом полученной невязки. Этот подход хорошо работает даже при большом количестве агентов.

В недавней работе [7] был предложен более сложный подход к многоагентному планированию, сочетающий безмодельное обучение с подкреплением и алгоритм MCTS. Авторы предложили схему из двух компонент, которая включает модуль достижения цели и модуль разрешения конфликтов. Последний был обучен с использованием MCTS. Построение дерева поиска для каждого из агентов также выполнялось независимо, а действия для других агентов выбирались с использованием текущей стратегии обучения. Примечательно, что в этой работе MCTS использовался исключительно на этапе обучения для построения стратегии разрешения конфликтов.

Алгоритм MATS-LP

Предлагаемый в данном параграфе метод MATS-LP сочетает в себе два основных ингредиента. Во-первых, в нем механизм MCTS используется для того, чтобы агент мог моделировать рассуждение о возможных будущих состояниях окружающей среды и выбирать наиболее перспективное действие для выполнения на текущем временном шаге, т.е. такое действие, которое, с одной стороны, максимизирует вероятность достижения цели (в конечном итоге) и, с другой стороны, уменьшает вероятность столкновений и взаимных блокировок с другими агентами. Во-вторых, на этапе симуляции в MCTS используется обучаемая стратегия. Эта стратегия аппроксимируется нейронной сетью и адаптирована для выполнения задач MAPF с точки зрения отдельного агента. Для предварительного предобучения такой стратегии используется известный метод обучения с подкреплением из семейства «актор-критик», предполагающий оптимизацию ближайшей стратегии (PPO) [512]. Необходимо отметить, что поскольку эта стратегия широко используется в MCTS для симуляции будущих состояний среды, она должна быть вычислительно эффективной (быстрой). На практике это означает, что нейронная сеть, которая аппроксимирует стратегию, должна содержать небольшое количество параметров (весов). Руководствуясь этим, было предложено использовать относительно компактную нейронную сеть, которая содержит 161 тыс. параметров по сравнению с миллионами параметров, настраиваемых в обычных современных аппроксиматорах стратегии. Например, количество параметров в одном из актуальных методов SCRIMP, с которыми проводится сравнение, составляет около 9 миллионов.

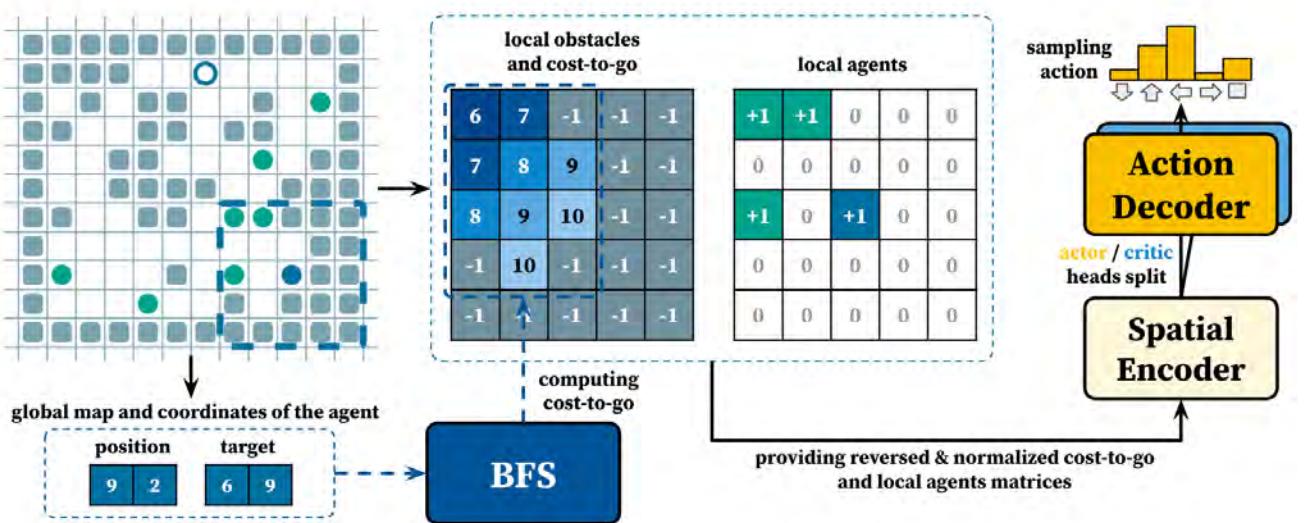


Рисунок 3.18 — На рисунке изображена схема алгоритма COSTTRACER. Подход использует две матрицы в качестве входных данных: одна кодирует препятствия, нормализованные обратные затраты на выполнение; другая содержит позиции локальных агентов. Весь конвейер обучается с помощью алгоритма PPO, используя функцию вознаграждения, которая обеспечивает положительную обратную связь только тогда, когда агент приближается к своей глобальной цели.

Многочисленные алгоритмы многоагентного обучения с подкреплением (MARL) могут быть использованы для решения задачи MAPF с частичной наблюдаемостью. Для интеграции обучения с подкреплением с алгоритмом MCTS в рассматриваемом случае наиболее подходящим является семейство методов «актор-критик», пример PPO [512], MAPPO [614] или FACMAC [257]. В проведенных экспериментах был использован алгоритм PPO, который формирует общую стратегию независимо для каждого агента.

В дополнение к выбору алгоритма необходимо определить пространство наблюдения и функцию вознаграждения, с помощью которой алгоритм будет обучаться в среде. В данном случае для построения графа сцены используется доступная локальная информация, аналогичная той, что используется в алгоритме PRIMAL2. Данное наблюдение предполагает, что агент располагает информацией о статических препятствиях на всей карте, знает свою текущую цель и может получать информацию о других агентах и их текущих целях в поле своего зрения.

Предлагаемый подход к безмодельному обучению называется COSTTRACER, что подчеркивает дизайн функции вознаграждения и входных данных для нейросетевого аппроксиматора. Он использует только две входные матрицы и простую функцию вознаграждения. Схематическое представление COSTTRACER показано на рисунке 3.18.

Наблюдение агента определяется как две матрицы размером $m \times m$. Первая матрица представляет позиции других агентов (+1, если агент присутствует, и 0, если его нет). Вторая матрица представляет нормализованную инвертированную функцию стоимости переходов (cost-to-go). Каждый раз, когда получен целевая позиция, функция стоимости переходов рассчитывается с использованием алгоритма поиска по ширине (BFS). Она

предоставляется агенту в нормализованном и инвертированном виде, то есть значение 1 в матрице соответствует ближайшей клетке к цели, видимой в пределах наблюдения агента. Препятствия представлены значением -1 , а все остальные значения находятся в диапазоне от 0 до 1.

Функция вознаграждения определяется следующим образом: агент получает вознаграждение в размере $+r$, если он достигает клетку ближе к цели в своей текущей истории эпизода. Расстояние до этой клетки обозначается как d^{+r} . Это расстояние измеряется по кратчайшему путем с использованием функции стоимости переходов и включается в центральную клетку наблюдения агента как $\min(1, \frac{d^{+r}}{64})$, гарантируя, что оно отмасштабировано в интервале $[0, 1]$. Во всех остальных случаях агент получает вознаграждение в размере 0. Эта функция вознаграждения обеспечивает плотный сигнал и предотвращает переобучение на ней, поскольку поведение, максимизирующее вознаграждение, по своей сути обеспечивает близость агента к цели.

Архитектура используемой нейронной сети включает три выхода: *пространственный кодировщик и декодировщик действий/полезностей* как для компонентов актора, так и для критика, в соответствии с подходом AlphaZero [423] (см. рисунок 3.18). Предлагаемая архитектура отличается тем, что в ней используется значительно меньшее количество параметров, чем в PRIMAL2 и SCRIMP, что позволяет обучить алгоритм на одном графическом процессоре менее чем за один час.

Несмотря на свою простоту по сравнению с другими современными алгоритмами, эта архитектура демонстрирует многообещающие результаты, как показано далее в экспериментальной части.

Схема многоагентного нейросетевого MCTS представлена на рисунке 3.19. Из-за того, что каждому агенту в среде доступна только частичная информация, использование централизованного планировщика невозможно. Чтобы иметь возможность планировать в таких ситуациях, предлагается использовать так называемый *внутренний МПР* (IMDP). Для этого создается внутренняя среда, основанная на эгоцентричном наблюдении агента (препятствия, другие агенты и их текущие цели). В эту среду включаются только те агенты, за которыми наблюдает текущий агент на текущем шаге. Все остальные клетки, которые не являются препятствиями, считаются пустыми.

Даже в этой внутренней среде количество агентов может быть значительным. Для решения этой проблемы был представлен метод маскирования действий, зависящий от близости других агентов к текущему агенту, для которого проводится планирование. Близость агентов устанавливается с использованием алгоритма BFS в пределах их наблюдений. Для первых K агентов, включая самого агента, рассматриваются все возможные действия, позволяющие избежать препятствий (недопустимые действия маскируются). И наоборот, для остальных агентов применяется «жадная» стратегия, согласно которой используется только действие с наибольшей вероятностью.

Пусть множество удаленных агентов обозначается как \mathbf{D} , а пространство действий этих агентов ограничивается одним действием с наибольшей вероятностью, обозначаемым как $A_{\mathbf{D}}^u = \arg \max_{a_u \in A} \pi(o_u, a_u)$ (предсказывается с помощью COSTTRACER). Конечное

количество переходов определяется путем перемножения немаскированных действий для каждого ближайшего агента.

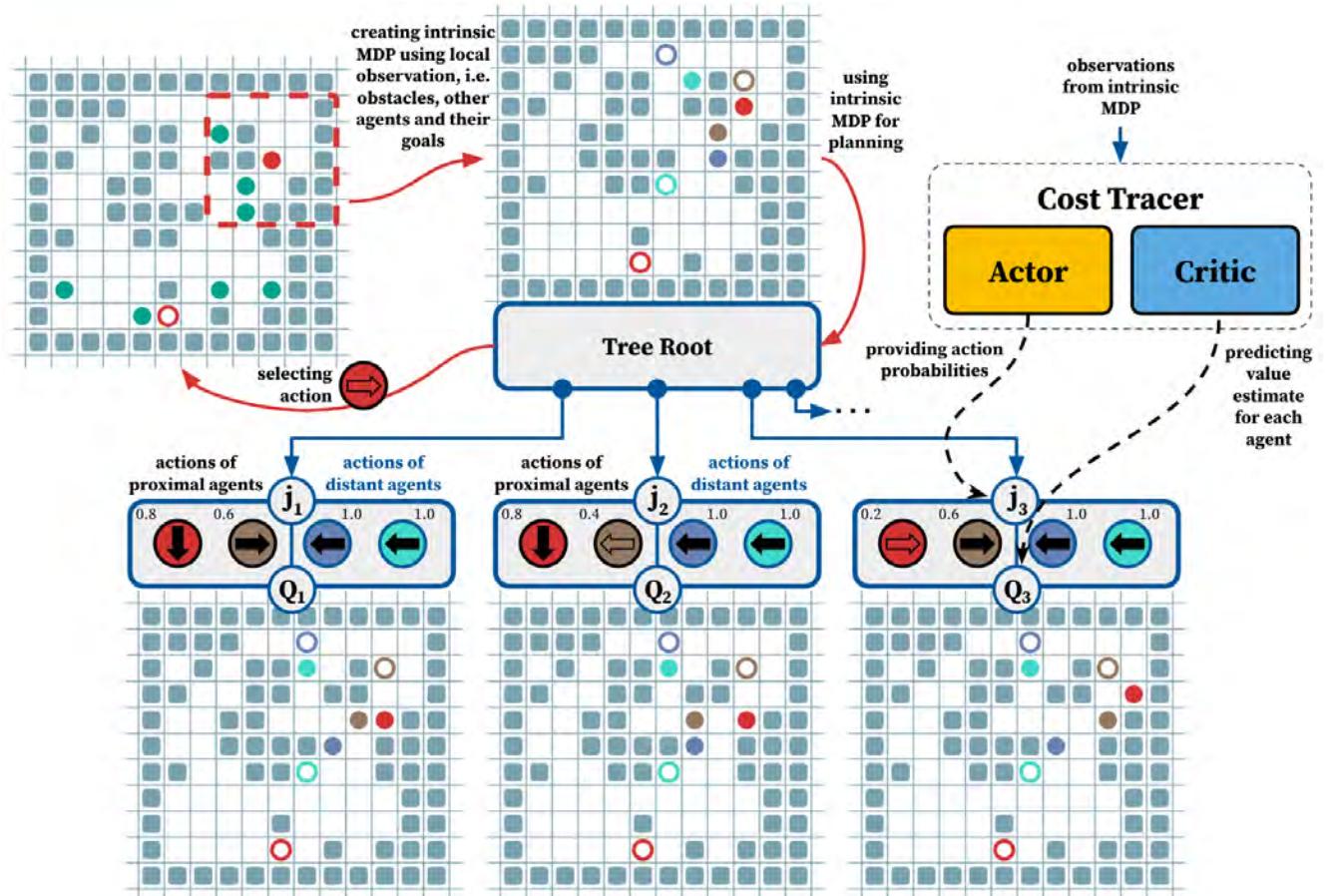


Рисунок 3.19 — Схема подхода MATS-LP. Сначала строится внутренний МППР (IMDP) с использованием глобальной карты статических препятствий, агентов в наблюдении и их текущих целей. Этот МППР служит основой для планирования с использованием MCTS. При таком подходе каждый узел дерева представляет совместное действие всех агентов, присутствующих в IMDP, вместе со статистикой связанных узлов. Вероятности действий и полезности узлов вычисляются с использованием алгоритма COSTTRACER. Необходимо отметить, что полезность для каждого агента вычисляется индивидуально, а оценка узла выводится как сумма полезностей по всем агентам. Процедура планирования концентрируется исключительно на агентах, находящихся в непосредственной близости. Например, в данном сценарии это включает в себя самого агента (выделен красным цветом) и коричневого агента. Это решение ограничивает рассмотрение ближайших агентов значительно упрощает процесс принятия решений. Действия агентов визуально представлены с помощью круговых иконок направлений, а жирные стрелки выделяют действия с наибольшей вероятностью. Для удаленных агентов рассматривается только одно действие с максимальной вероятностью.

Во время прямого поиска в таком МППР максимизируется совместное вознаграждение в размере r для всех агентов. Функция вознаграждения IMDP идентична функции вознаграждения для COSTTRACER. Каждый узел в дереве поиска соответствует внутреннему состоянию s IMDP. Для каждого совместного действия j из состояния s устанавливается

ребро (s, \mathbf{j}) для хранения набора статистических данных $\{N(s, \mathbf{j}), Q, r, \pi_{\mathbf{j}}\}$. Здесь N представляет количество посещений узла, Q — среднее совместная полезность действия, r — совместное вознаграждение, полученное от IMDP при выполнении действия \mathbf{j} , а π_j обозначает вероятность совместного действия \mathbf{j} . Примечательно, что здесь используется обозначение s для состояния IMDP.

Процесс поиска в таком IMDP, как и для MCTS, состоит из трех отдельных этапов (за исключением симуляции):

Выбор. Выбор узла начинается с корня дерева s^0 , который является начальным состоянием IMDP. Процесс выбора продолжается до тех пор, пока не будет достигнут конечный узел, который обозначается как s^l , где l представляет продолжительность одной итерации прямого поиска. Каждое действие выбирается на основе статистики, хранящейся в узлах дерева. Эта процедура выполняется в соответствии с формулой вероятностного UCT (PUCT), введенной в [420]:

$$\mathbf{j}^k = \arg \max_{\mathbf{j}} \left(Q(s, \mathbf{j}) + c \pi_j \frac{\sqrt{\sum_{\mathbf{i}} N(s, \mathbf{i})}}{1 + N(s, \mathbf{j})} \right).$$

Здесь \mathbf{i} представляет все возможные совместные действия из текущего узла, а $\pi_j = \prod_u \pi_u(s, \mathbf{j})$ — вероятность совместного действия. Константа c контролирует влияние распределения стратегий на Q . Переход к следующему состоянию внутренней среды осуществляется путем применения в ней действия \mathbf{j} . π_j и оценка полезности узла $v^l = \sum_u v(o_u, \mathbf{j})$ вычисляются с использованием COSTTRACER, а вознаграждение r накапливается с использованием сигнала, предоставляемого IMDP.

Расширение. На конечном временном шаге l создается новый узел. Переход к следующему состоянию внутренней среды осуществляется путем применения действия \mathbf{j}^l . Вероятности действий $\pi_u(s^l, \mathbf{j}^l)$ рассчитываются с использованием COSTTRACER, а вознаграждение для каждого агента $R(s, u, \mathbf{j})$ накапливается с использованием сигнала, предоставляемого IMDP. Статистика нового узла инициализируется следующим образом: $N^l(s^l, \mathbf{j}^l) = 0, Q^l = 0, r = \sum_u R(s, u, \mathbf{j}), \pi_j^l = \prod_u \pi_u^l(o_u^l, a_u^l)$.

Обратное распространение. Это заключительный шаг, на котором обновляется накопленная статистика по траектории. Обновление вычисляется с использованием коэффициента дисконтирования γ , аналогично безмодельному обучению с подкреплением. Для формирования оценки совокупного дисконтированного вознаграждения за траекторию, используется выражение:

$$G^k = \sum_{\tau=0}^{l-1-k} \gamma^\tau r_{k+1+\tau} + \gamma^{l-k} v^l.$$

После этого статистика для каждого ребра (s^{k-1}, \mathbf{j}^k) обновляется следующим образом:

$$Q(s^{k-1}, \mathbf{j}^k) := \frac{N(s^{k-1}, \mathbf{j}^k) + Q(s^{k-1}, \mathbf{j}^k) + G^k}{N(s^k, \mathbf{j}^k) + 1},$$

$$N(s^{k-1}, \mathbf{j}^k) := N(s^{k-1}, \mathbf{j}^k) + 1.$$

Итоговое действие, выполняемое агентом в среде, определяется как действие, относящееся к наиболее исследованному ребру из корня дерева, определяемое в свою очередь количеством посещений $N(s, j)$. Действие a_u агента, для которого был создан IMDP, берется из j . Окончательное совместное действие в глобальной среде формируется как действия всех эгоцентрических агентов, запланированные с помощью MCTS в их IMDP. После выполнения этого действия в среде каждый агент получает свои локальные наблюдения, заново воссоздает свой IMDP, и процесс повторяется.

Экспериментальное исследование алгоритма MATS-LP

Для оценки эффективности алгоритма MATS-LP был проведен ряд экспериментов, позволяющий сравнить его с существующими обучаемыми подходами, адаптированными для решения задач LMAPF. Во всех экспериментах продолжительность эпизода была установлена равной 512. Все агенты имели одинаковые параметры: их наблюдение было размером 11×11 клеток, все возможные действия рассматривались только для ближайших 3 агентов, включая основного агента, значение γ было установлено равным 0.96, количество этапов расширения за итерацию — 250, коэффициент c был установлен равным 4,4.

Для сравнения были выбраны два других обучаемых подхода: современный метод решения многоцелевого MAPF – PRIMAL2 [507] – и недавно представленный метод, который показал впечатляющие результаты при решении одноцелевого MAPF – SCRIMP [556]. Согласно результатам, представленным в оригинальной статье SCRIMP, он явно превосходит некоторые другие существующие подходы по решению классической задачи MAPF: PICO [450] и DHC [414]. В связи с этим последние два алгоритма не рассматривались в качестве конкурентных базовых решений.

Была использована реализация и веса сети, предоставленная авторами PRIMAL2³ и SCRIMP⁴. Код SCRIMP был адаптирован для решения многоцелевой задачи MAPF, а именно была использована локальная версия SCRIMP, которая имеет ограниченный радиус коммуникации агентов (5) и которая показывает лучшие результаты. Размер наблюдения для SCRIMP был установлен равным 3×3 , в то время как для PRIMAL2 — 11×11 .

Сравнение проводилось на трех типах карт с разной топологией. Первый тип состоит из клеточных карт размером 20×20 со случайно расположенными препятствиями. Плотность препятствий варьируется от 0% до 30%. Всего было использовано 40 случайных карт. Для каждой карты было сгенерировано 5 различных экземпляров со случайным расположением начальной клетки и цели. Второй тип карт — это лабиринтоподобные среды, которые были сгенерированы с помощью генератора из репозитория PRIMAL2. Были сгенерированы лабиринты с размерами 10×10 , 20×20 и 30×30 , по 50 карт на каждый размер, один (случайно сгенерированный) экземпляр задачи на каждую карту. Наконец, для оценки была

³<https://github.com/marmotlab/PRIMAL2>

⁴<https://github.com/marmotlab/SCRIMP>

использована карта склада размером 33×46 из работы [408]. 10 случайных экземпляров на этой карте были сгенерированы и использованы для оценки.

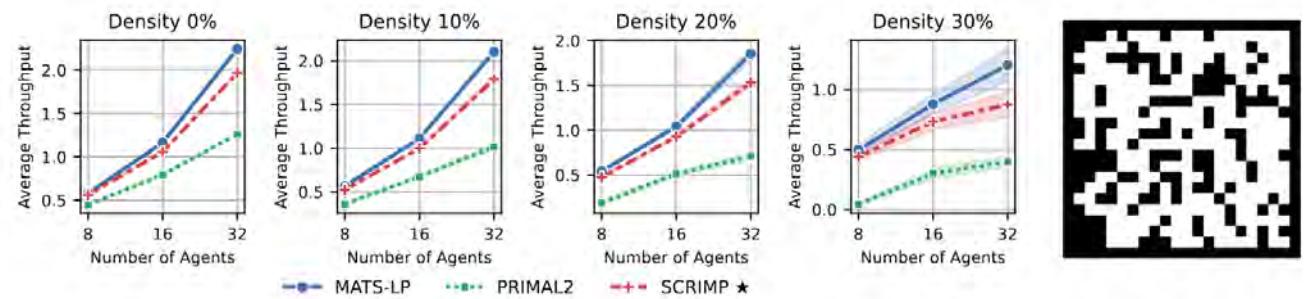


Рисунок 3.20 — Средняя пропускная способность (throughput) MATS-LP, SCRIMP и PRIMAL2 на случайных картах 20×20 с различной плотностью препятствий. Символом \star отмечены подходы, которые были обучены на картах соответствующего типа.

Заштрихованные области указывают на 95% доверительные интервалы.

Для обучения алгоритма COSTTRACER была использована асинхронная реализация алгоритма PPO с открытым исходным кодом⁵. Был использован кодировщик наблюдения ResNet в качестве модуля *пространственного кодировщика* с одним остаточным (residual) слоем, а размеры скрытых слоев для блоков многослойного перцептрона (MLP) были равны 32 для *декодировщика действий/полезностей*. В процессе обучения коэффициент дисконтирования (γ) составлял 0,96, а скорость обучения — 0,00013.

Был использован байесовский гиперпараметрический поиск для оптимизации параметров алгоритма и нейросетевой архитектуры. В общей сложности было проведено 100 запусков алгоритма, что примерно соответствует 120 часам работы графического процессора с использованием одного Titan RTX. Была выбрана модель, показавшая наилучшие результаты при меньшем количестве параметров.

Результаты на случайных картах представлены на рисунке 3.20. В этом эксперименте MATS-LP превосходит SCRIMP во всех случаях, увеличивая пропускную способность в среднем на 15,6%. В то же время PRIMAL2 демонстрирует низкую производительность при более чем в два раза меньшей пропускной способности, чем MATS-LP в среднем. Такое поведение объясняется тем, что PRIMAL2 адаптирован для решения задачи MAPF на картах, состоящих из коридоров, таких как лабиринтные среды. Более того, он был обучен на лабиринтоподобных картах, подобных MATS-LP. Таким образом, этот тип карты не входит в обучающее распределение карт для этих двух подходов. Заштрихованные области на рисунке указывают на 95%-ные доверительные интервалы. Детальный анализ результатов показал, что пропускная способность может значительно варьироваться от карты к карте, поскольку некоторые карты содержат узкий проход, разделяющий среду на две части, и многие агенты не достигают целей, пытаясь пройти через проход в противоположных направлениях, блокируя друг друга.

Результаты второй серии экспериментов на лабиринтоподобных картах различных размеров представлены на рисунке 3.21. Как и в первой серии экспериментов, MATS-LP

⁵<https://github.com/alex-petrenko/sample-factory>

значительно превосходит обоих конкурентов. По сравнению с алгоритмом SCRIMP он показал в среднем на 46,2% более высокую пропускную способность, в то время как PRIMAL2 он превзошел на 38,8%. В большинстве случаев SCRIMP и PRIMAL2 демонстрируют в среднем почти одинаковую эффективность, за исключением карт лабиринта размером 30×30 с количеством агентов 32 или 64, где PRIMAL2 существенно превзошел SCRIMP, продемонстрировав немного лучшую масштабируемость на картах такого типа.

В последней серии экспериментов использовалась карта типичного складского помещения, которая была взята из работы [408]. Был использован тот же способ генерации начальных и конечных местоположений для агентов, что и в оригинальной статье, когда начальные местоположения для всех агентов могли располагаться только на левом или правом краю карты, в то время как конечные местоположения — только рядом с препятствиями в середине карты. Из-за ограничений, наложенных на возможные места расположения агентов, общее количество агентов на этом типе карты не может превышать 192. В соответствии с этими ограничениями было сгенерировано 10 различных экземпляров этой карты.

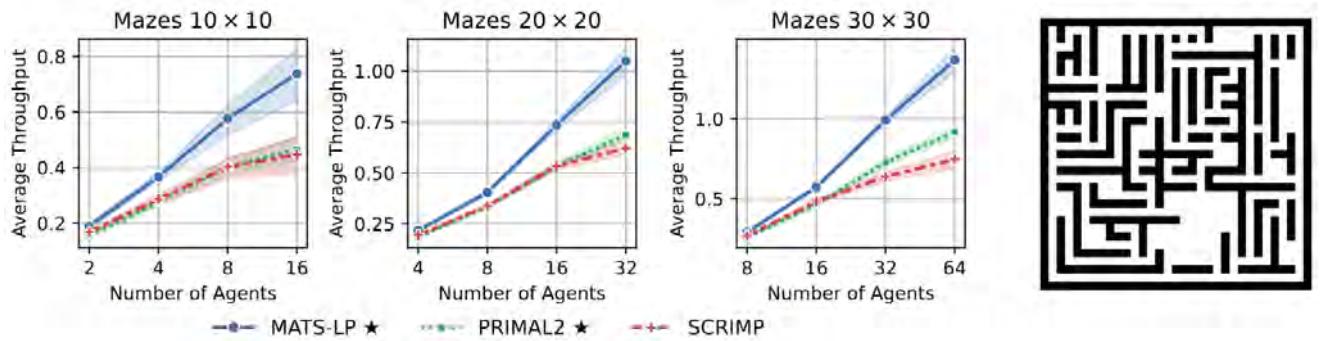


Рисунок 3.21 — Средняя пропускная способность MATS-LP, SCRIMP и PRIMAL2 на лабиринтоподобных картах различных размеров. Символом \star отмечены подходы, которые были обучены на картах соответствующего типа.

Результаты этих экспериментов показаны на рисунке 3.22. В дополнение к измерению средней производительности подходов, было также оценено время, необходимое для принятия решения о следующем действии для каждого агента, и проведено исследование влияния различных компонент алгоритма MATS-LP на общую производительность.

Левый график рисунка 3.22 показывает усредненную пропускную способность. Здесь также наблюдается, что MATS-LP демонстрирует лучшую производительность, его пропускная способность на 15,8% выше, чем у SCRIMP (в среднем), и на 27,1% выше, чем у PRIMAL2.

Средний график демонстрирует время, необходимое каждому из решателей LMAPF для принятия решения о следующем действии для одного агента (время принятия решения). Был также добавлен график для COSTTRACER, обучаемой стратегии, используемой в MATS-LP. Очевидно, что его время принятия решения практически постоянно, в то время как время MATS-LP увеличивается со 103 мс (миллисекунд) до 300 мс, когда число агентов увеличивается с 32 до 192. Это может быть объяснено увеличением числа агентов, которые

появляются в поле зрения каждого агента, и тем фактом, что MATS-LP предсказывает действия этих наблюдаемых агентов, что требует существенных затрат времени (поскольку требует более частого запуска COSTTRACER). Аналогичным образом время принятия решения для SCRIMP непостоянно и колеблется от 25 до 107 мс из-за необходимости координации действий с большим количеством агентов. Хотя MATS-LP требует больше времени, чем SCRIMP, для принятия решения, его масштабируемость немного лучше, т.е. отношение во времени принятия решения между 192 агентами и 32 агентами составляет коэффициент 3 для MATS-LP и коэффициент 4 для SCRIMP.

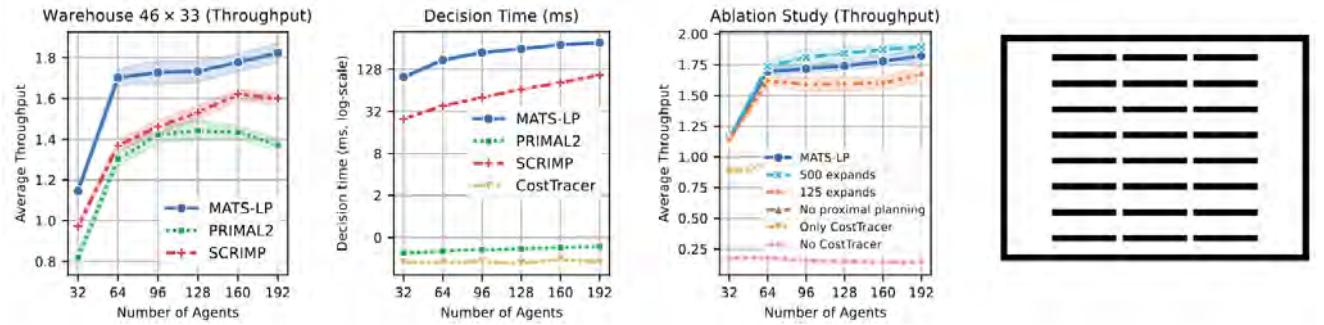


Рисунок 3.22 — Средняя пропускная способность и среднее время принятия решения MATS-LP, SCRIMP и PRIMAL2, а также исследование влияния компонент алгоритма MATS-LP на складской карте. Заштрихованные области указывают на 95% доверительные интервалы.

Правый график на рисунке 3.22 демонстрирует результаты исследования влияния различных компонент алгоритма MATS-LP. Здесь COSTTRACER — это MATS-LP с отключенным поиском MCTS. Была также проведена оценка алгоритма MATS-LP со случайной стратегией вместо COSTTRACER. Термин «No proximal planning» относится к варианту, в котором планирование выполняется исключительно для эгоцентричного агента за счет выбора только действия с наибольшей вероятностью для других агентов. Кроме того, был проведен эксперимент с увеличением количества шагов расширения до 500 и сокращением их до 125 по сравнению с количеством этих шагов 250, используемым в базовой версии MATS-LP. Худшие результаты демонстрирует версия, использующая случайную стратегию вместо COSTTRACER, что указывает на ее критическую важность. Следующая по величине пропускная способность получена COSTTRACER, пропускная способность которого почти в два раза хуже по сравнению с MATS-LP. Результаты версии, которая планирует только для эгоцентричного агента, ухудшаются с увеличением числа агентов, поскольку увеличение плотности агентов увеличивает необходимость координации между ними. Версии с увеличенным/уменьшенным количеством шагов расширения дерева поиска показывают результаты немного лучше или хуже, чем базовая версия соответственно. Последнее указывает на то, что, хотя MATS-LP имеет относительно высокое время принятия решения, на самом деле его можно настроить на требуемый бюджет по времени или даже работать за любое необходимое время, адаптируясь к определенному временному бюджету.

В табл. 6 представлены гиперпараметры алгоритмов COSTTRACER и MATS-LP. Здесь параметр «Number of agents» обозначает количество агентов в среде, в которой был обучен

COSTTRACER. Параметры таблицы, помеченные как «Tuned», были оптимизированы с использованием байесовского поиска. Для других параметров были использованы значения по умолчанию, обычно используемые в других работах по обучению с подкреплением. Параметр «Root exploration ratio» соответствует значению шума (с равномерным распределением), добавленному в корень дерева для облегчения поиска в алгоритме MCTS.

В данном разделе диссертационного исследования была рассмотрена многоцелевая задача MAPF и предложено ее решение, основанное на поиске по дереву Монте-Карло, интегрированном с облегченной обучаемой стратегией, предобученной для решения эгоцентричной задачи MAPF. Получаемый в итоге решатель децентрализован и не требует проработки явного взаимодействия с другими агентами. Эмпирически было показано, что предложенный решатель может хорошо обобщаться на экземпляры задачи LMAPF, которые он не встречал в процессе обучения, и может превосходить самых современных конкурентов в различных сложных условиях. Важным направлением будущих исследований этой части диссертационного работы является разработка полностью обучаемого алгоритма MCTS для задачи LMAPF, т.е. с обучением стратегии симуляции с помощью самого поиска по дереву, как, например, в работе [420].

3.4 Исследование среды в обучении с подкреплением на основе модели

В данном разделе диссертационного исследования рассмотрена реализация обучения стратегии на основе модели в архитектуре NSLP с акцентом на особенности реализации механизмов исследования среды. Предложена общая концепция интеграции дополнительных сигналов вознаграждения в процесс обучения латентной модели мира M_L и при обучении актора и критика. В данном разделе предполагается, что агенту доступно исходное сенсорное представление наблюдения без объектной декомпозиции. Основные результаты, изложенные в разделе, представлены в публикациях [87].

Как уже неоднократно упоминалось ранее, в обучении с подкреплением разрабатывается ряд методов [323; 56], использующих обучаемую модель мира, которая позволяет агенту сохранять и обобщать знания о динамике взаимодействий в среде. Обучение модели является активным, то есть опирается на опыт взаимодействия агента со средой, что, в свою очередь, имеет как достоинства, так и недостатки. С одной стороны, серьезной проблемой является изменение поведения по мере обучения агента, которое становится все более направленным на решение поставленной задачи, что затрудняет использование модели в решении других задач даже из данного класса (проблема обобщающей способности). С другой стороны, определенная корректировка стратегии поведения агента позволяет изменить обучающую выборку для увеличения репрезентативности и полноты модели.

Таблица 6 — Параметры алгоритмов COSTTRACER и MATS-LP. В столбце «Tuned» указаны параметры, которые были оптимизированы с помощью гиперпараметрического поиска.

COSTTRACER	Value	Tuned
Adam learning rate	0,000 13	✓
γ (discount factor)	0.96	✓
PPO clip ratio	0.2	
PPO optimization epochs	1	✓
Batch size	2048	✓
Entropy coefficient	0,068 78	✓
GAE_λ	0.95	
ResNet residual blocks	2	✓
ResNet number of filters	32	✓
Activation function	ReLU	
Network initialization	orthogonal	
MLP size	32	✓
Number of agents	[64, 128]	✓
Parallel environments	16	
Training steps	75 000 000	
Observation patch	11×11	
Network parameters	161 734	

MATS-LP	Value	Tuned
Discount factor γ	0.96	
Exploration coefficient c	4.4	✓
Number of expansions	250	
Planning agents K	3	✓
Root exploration ratio	0.6	✓

Реализацию механизма улучшения данного подхода подсказывают механизмы внутренней мотивации человека [541], которые описывают, как происходит обучение в отсутствии конкретной задачи, то есть предлагают способы автономного обучения. Подходы к использованию внутренней мотивации для построения интеллектуальных агентов образуют отдельное направление [138; 139; 333; 486], в котором ключевым является сигнал внутреннего вознаграждения — аналог вознаграждения в обучении с подкреплением.

Механизмы внутренней мотивации рассматривают три способа изменения поведения агента. Самым распространенным является корректировка вознаграждения, где сигнал внутреннего вознаграждения смешивается со стандартным вознаграждением. Другим

вариантом является введение дополнительной стратегии для исследования среды, которая обучается на основе чистого сигнала внутреннего вознаграждения. Третий способ заключается в построении вспомогательных исследовательских подзадач (постановка внутренних целей на основе сигнала внутреннего вознаграждения), решение которых обеспечивает агента универсальными навыками для данной среды.

В данном разделе рассматриваются современные работы в области обучения с подкреплением в контексте формирования модели мира и применения механизмов внутренней мотивации, использующих модель мира, для улучшения эффективности автономного поведения агента. Задача данного раздела исследования — предложить единый подход, систематизирующий накопленные знания в данном направлении; наметить перспективные пути дальнейшего развития области.

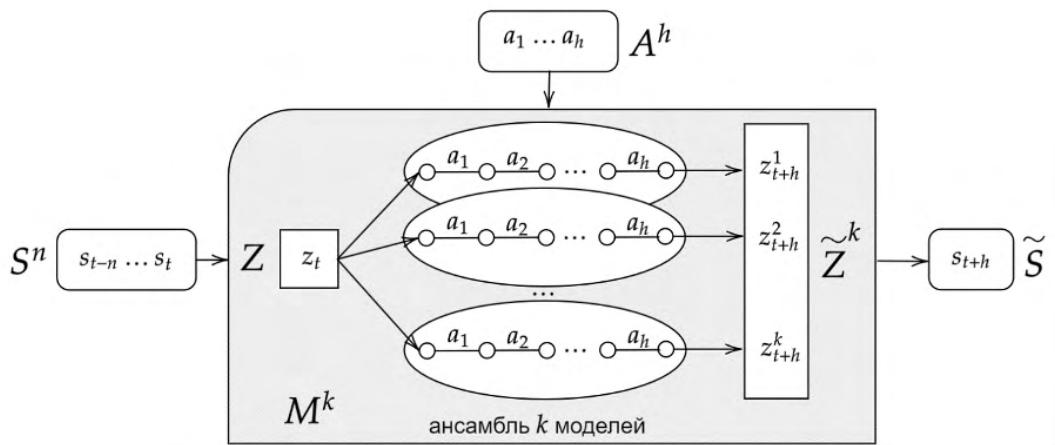


Рисунок 3.23 — Возможные способы связи состояний и действий в модели. S^n — контекст из n состояний. Z — латентное пространство представлений. M^k — ансамбль из k отдельных моделей. A^h — последовательность из h действий. \tilde{S}, \tilde{Z} — состояние и представление, в котором окажется агент через h шагов от начального.

Существует множество способов формального описания аппроксиматора динамики среды [443]. Ключевым отличием каждого является метод определения взаимоотношений между состоянием S , выполненным действием A , и следующим состоянием \tilde{S} (см. рисунок 3.23). Модель M связывает состояния после выполнения одного действия ($h = 1$) или нескольких. Сама связь может быть как детерминированной, так и вероятностной, когда модель описывает распределение возможных следующих состояний для конкретного текущего состояния и действия. При отсутствии в среде марковского свойства модель может выстраивать собственное пространство представлений Z , где каждое из представлений определяется некоторым предыдущим контекстом S^n . Также широко распространено использование не одной модели M [201], а сразу ансамбля M^k [213; 227; 491; 566], в котором каждая отдельная часть выступает как независимая модель, выдающая свое предсказание.

3.4.1 Внутренняя мотивация в процессе обучения

В психологии [541] внутренняя мотивация определяется как выполнение активности в силу ее собственной привлекательности, а не получаемых результатов. Внутренне мотивированный человек получает удовольствие от самой деятельности без внешних стимулов, что противоположно внешней мотивации, которая полностью определяется внешним влиянием: похвала учителя, денежное вознаграждение, страх наказания.

В обучении с подкреплением, основная идея которого основана на сигнале поощрения и наказания, изначально определена именно внешняя мотивация. Однако было показано [145], что механизмы внутренней мотивации естественно описываются в формализме вычислительного обучения с подкреплением: для перехода к внутренне мотивированному варианту достаточно вместо стандартного внешнего вознаграждения R использовать специальную функцию внутреннего вознаграждения R_{int} . R_{int} зависит не только от перехода в среде $\langle s_t, a_t, s_{t+1} \rangle$, как функция R , но и от внутренних представлений Z и модели M , то есть опыта взаимодействия со средой. Замена функции вознаграждения меняет задачу, которую решает агент. Возникает вопрос: «Как применить агента на основе внутреннего вознаграждения для повышения эффективности нахождения решения исходной задачи с вознаграждением R ?».

Выделяют три основных способа решения этой проблемы. В самом распространенном предлагается использовать в качестве функции вознаграждения агента некоторую комбинацию R и R_{int} . Во втором объединение происходит на уровне стратегии. Обучаются две раздельные стратегии: одна на основе R , другая на основе R_{int} , — а затем эти стратегии совмещаются в конечную, определяющую поведение агента. Наконец, в третьем варианте к целям агента (цели предполагаются заданными в явном виде) добавляются цели, сгенерированные на основе внутреннего вознаграждения. На рисунке 3.24 отображены уровни агента, на которых происходит реализация каждого из способов, а в табл. 7 отображены способы, используемые в существующих алгоритмах обучения с подкреплением.

В общем виде обучение внутренне мотивированного агента состоит из обучения его модели и двух стратегий. Этот процесс является активным в том смысле, что, выбирая определенные действия в текущем состоянии, агент в результате получает обучающие примеры, которые откладываются в памяти D — источнике обучающей выборки. Задача обучения формально определяется через оптимизацию некоторого функционала (см. раздел 2.2) для такой выборки.

Для каждой из обучаемых компонент агента: целевой стратегии π_g , исследовательской стратегии π_ϵ и модели M — в общем случае присутствует своя память (обучающая выборка) соответственно: D_g, D_ϵ, D_M (см. рисунок 3.24 и 3.25). Память представляет набор траекторий агента $\tau_H = (s_i, a_i, s_{i+1})_{0:H-1}$, состоящих из H последовательных переходов из одного состояния s_i в другое s_{i+1} под действием a_i . Действие выбирается согласно стратегии; следующее состояние определяется из динамики переходов.

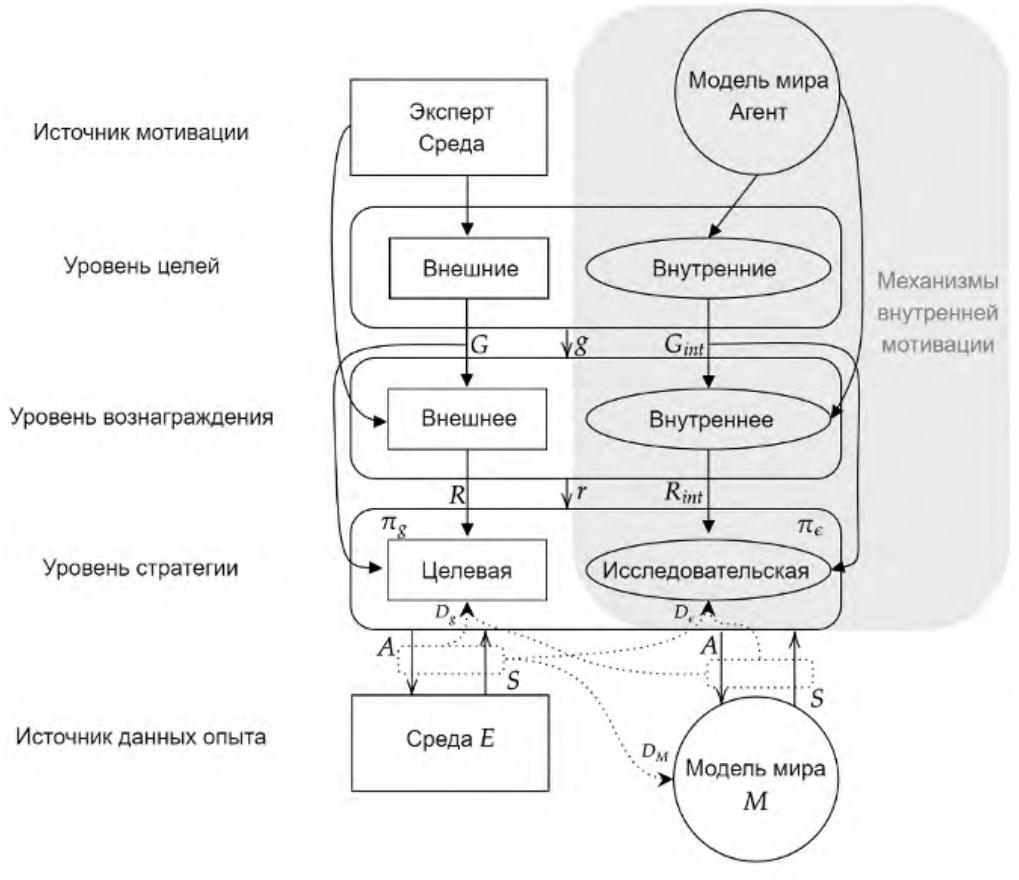


Рисунок 3.24 — Уровни внутренне мотивированного агента. На каждом из уровней методы внутренней мотивации предлагают свой аналог: стратегии, вознаграждения или цели.

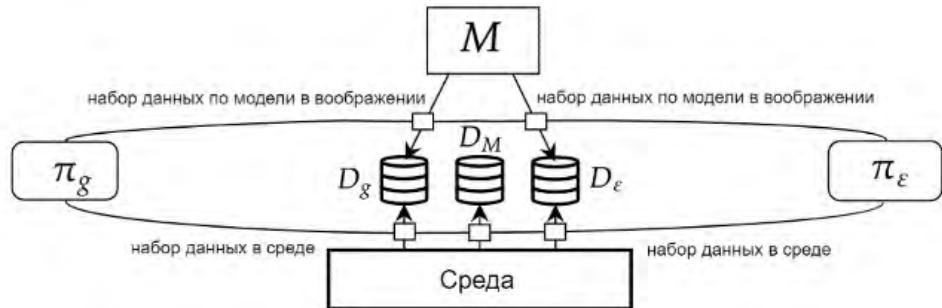


Рисунок 3.25 — Сбор данных для обучения. Целевая и исследовательские стратегии собирают данные для обучения из модели мира и среды в память D_g , D_ϵ , D_M .

Для заполнения памяти действия агента выполняются либо в среде (примеры переходов из истинной динамики МППР), либо в «воображении» (когда переходы определяются обучаемой моделью мира), либо смешано (т.е. какая-то доля траекторий определяется моделью, а другая доля средой):

$$\begin{aligned} D_g &= \{\tau_h = (s_i, a_i, s_{i+1})_{0:H-1} | a_i \sim \pi_g(s_i), s_{i+1} \sim [M, T](s_i, a_i)\}, \\ D_\epsilon &= \{\tau_h = (s_i, a_i, s_{i+1})_{0:H-1} | a_i \sim \pi_\epsilon(s_i), s_{i+1} \sim [M, T](s_i, a_i)\}. \end{aligned} \quad (3.4)$$

Важным отличием памяти, используемой для обучения модели, являются действия, определяемые любой из стратегий, но только для взаимодействия с истинной динамикой:

$$D_M = \{\tau_h = (s_i, a_i, s_{i+1})_{0:H-1} | a_i \sim [\pi_g, \pi_\epsilon](s_i), s_{i+1} \sim T(s_i, a_i)\}. \quad (3.5)$$

Таблица 7 — Характеристика рассматриваемых методов внутренней мотивации по наличию соответствующих компонент на разных уровнях агента (вознаграждения, стратегии, цели, см. рисунок 3.24) и типу сигнала внутренней мотивации.

Метод	R_{int}	π_ϵ	G_{int}	Тип сигнала
SelMo [335]	—	—	+	$L[M]$
ICM [201]	+	—	—	$L[M]$
EMI [246]	+	—	—	$L[M]$
Plan2Explore [499]	—	+	—	$L[M]$
LEXA [227]	—	+	+	$L[M]$
MEEE [543]	+	—	—	$L[M]$
MAX [566]	—	+	—	$L[M]$
AWML [117]	—	+	—	ΔM
VIME [637]	+	—	—	ΔM
Deep ICAC [213]	+	—	—	ΔM
GDE [642]	+	—	—	χM
WRW [643]	—	+	+	χM
EC [253]	+	—	—	χM
CEE-US [548]	—	+	—	$L[M]$
Director [212]	—	—	+	$L[M]$
CC-RIG [194]	—	—	+	χM^*
SMORL [655]	—	—	+	χM^*
SRICS [654]	—	—	+	χM^*

Особенности стратегий вносят свой вклад и в специфику данных, накапливаемых в памяти. Основной задачей целевой стратегии является решение МППР, поставленного для определенной цели, из-за чего накапливаемый опыт коррелирует с целью. Исследовательская стратегия направлена на получение наибольшего количества информации и, соответственно, обеспечивает большее разнообразие переходов в траекториях, что улучшает обучение модели (более универсальные данные).

При формальной постановке задачи обучения определяется оптимизационная задача по нахождению минимума некоторого функционала, определяемого через функцию потерь L . Каждая из обучаемых компонент агента обладает своим функционалом L_g, L_ϵ, L_M , которые либо оптимизируются независимо, либо рассматриваются совместно.

Целевая стратегия для обучения требует траекторий, а также соответствующих им значений сигнала вознаграждения. Вознаграждение определяется (см. рисунок 3.24) на основе полностью внешней мотивации, либо с использованием внутреннего вознаграждения. Помимо вознаграждения, содержащего информацию о цели, стратегия $\pi_g(a|s,g)$ может явно зависеть от цели. Функционал целевой стратегии:

$$L_g = \mathbb{E}_{\tau_H \sim D_g} l(\tau_h, r, g, \pi_g), \quad (3.6)$$

где D_g — память, g — цель, $r = [R, R_{int}]$ — объединенный сигнал вознаграждения, а l — одна из стандартных для обучения с подкреплением, например, дисконтированная отдача.

Решая похожую оптимизационную задачу, исследовательская стратегия, в отличие от целевой, опирается исключительно на внутреннее вознаграждение. Такая стратегия не зависит от цели и является универсальной для среды. Ее функционал:

$$L_\varepsilon = \mathbb{E}_{\tau_H \sim D_\varepsilon} l(\tau_H, R_{int}, \pi_\varepsilon). \quad (3.7)$$

Для модели мира процесс обновления отличается от обучения стратегий, так как для нее решается задача обучения с учителем. В данном случае уже известны правильные примеры, которые были собраны стратегиями в среде. Необходимо научиться их воспроизводить. Функции потерь для модели могут быть разнообразными, но основным является вычисление разности между предсказаниями модели и примерами из обучающей выборки:

$$\begin{aligned} L_M &= \mathbb{E}_{\tau_H \sim D_M} l_M(\tau_H, M), \\ L_M &= \mathbb{E}_{\tau_H \sim D_M} l_F(M(S^n, A^h), \tilde{S}), \\ L_M &= \mathbb{E}_{\tau_H \sim D_M} l_I(M(S^n, \tilde{S}), A^h), \end{aligned} \quad (3.8)$$

где S^n, A^h, \tilde{S} — состояние, действия и предсказанные состояния (см. рисунок 3.23) из памяти D_M , l_M — функция, определяющая разницу между двумя объектами: предсказанием и истиной (например, для векторов сумма квадратов разницы), а l_I и l_F — функционалы модели обратной и прямой динамики соответственно.

3.4.2 Формирование сигнала внутренней мотивации

Методы внутренней мотивации позволяют корректировать обучение агента и его модели мира, используя накопленные знания агента о динамике среды. Модель мира — источник внутренней мотивации (см. рисунок 3.24) — снабжает информацией, необходимой для формирования внутреннего вознаграждения R_{int} , обеспечивает возможность обучения исследовательской стратегии π_ε в «воображении» (без взаимодействия с реальной средой), представляет динамику мира в удобной форме (например, в виде графа) для выделения важных состояний, которые становятся внутренними целями.

У каждого уровня агента (см. рисунок 3.24) своя специфика взаимодействия с методами внутренней мотивации на основе модели, однако везде используется сигнал внутренней мотивации, который вводился как внутреннее вознаграждение.

Для обучения стратегии агента необходим сигнал вознаграждения (обратной связи), который численно характеризует четверку $\langle s_t, a_t, s_{t+1}, g \rangle$. С точки зрения методов внутренней мотивации, выделяют сигналы вознаграждения: основанные на знаниях (knowledge-based [333]), которые зависят только от состояний и действий $\langle s_t, a_t, s_{t+1} \rangle$, и

основанные на умениях (competence-based [333]), которые характеризуют внутренние цели и возможность их достижения. С точки зрения определения сигнала внутренней мотивации на основе модели, такое разделение методов не отображает специфики, характерной для модельного подхода. При обучении модели мира действия агента обеспечивают поиск информации о реальной динамике среды. Однако агенту необходимы маркеры, обозначающие успешность процесса исследования. Такие маркеры – сигналы внутренней мотивации – используют

- текущую неопределенность модели, т. е. некоторый вид ошибки;
- изменение накопленной в модели информации за счет полученных данных;
- морфологию среды, позволяющую выявить важные переходы, действия или состояния.

Неопределенность модели $L[M]$ самая обширная группа способов определения сигнала внутренней мотивации (см. табл. 7):

$$L[M] : R_{int} = \begin{cases} |\tilde{S} - M(S)|, \\ D[M^k(S)], \end{cases} \quad (3.9)$$

где $|\tilde{S} - M(S)|$ в общем виде обозначает невязку модели, а $D[M^k(S)]$ – разброс предсказаний ансамбля моделей (например, дисперсия). Этот сигнал позволяет направить исследование на выполнение действий, последствия которых агент ещё недостаточно хорошо изучил (высокая неопределенность модели). Такое взаимодействие генерирует опыт, позволяющий улучшить модель мира агента в проблемных местах. Один из сигналов неопределенности модели 3.9 – ошибка предсказания следующего состояния (ICM [201], SelMo [335], EMI [246], Director [212]). Однако для его вычисления необходим постоянный доступ к истинному состоянию, которое должно быть следующим.

Другим вариантом сигнала из этой группы является неопределенность предсказаний ансамбля моделей. Этот сигнал 3.9 позволяет освободиться от проблемы обязательного получения истинного следующего состояния для вычисления сигнала вознаграждения. Такая его особенность важна в случае, когда обучение строится на основе опыта, полученного из взаимодействия с моделью. Предсказания ансамбля моделей используются в таких алгоритмах, как Plan2Explore [499], LEXA [227] и MEEE [543], а алгоритм MAX [566] использует ансамбль моделей, при этом для описания разброса предсказаний использует дивергенцию Йенсена-Шеннона.

Пополнение знаний ΔM – сигналы мотивации, определяющие изменение модели при добавлении к ней новой информации:

$$\Delta M : R_{int}(t) = |M(t) - M(t - n)|, \quad (3.10)$$

где $M(t)$ – состояние модели в момент времени t , а n – горизонт временных шагов, за который отслеживается изменение модели. Примеры таких сигналов: изменение функции потерь за несколько шагов (AWML γ -Progress [117]), дивергенция Кульбака-Лейблера между предсказаниями текущей и обновленной модели (VIME [637]), изменение предсказания ансамбля (Deep ICAC [213]).

Морфология среды $\chi[M]$ определяет набор сигналов вознаграждения, характеризующий структурные свойства мира:

$$\chi[M] : R_{int} = X[M, S], \quad (3.11)$$

где X — некий функционал, позволяющий численно охарактеризовать одно из морфологических свойств среды. Такой сигнал прямо или явно не связан ни с обучением стратегии, ни с обучением модели мира, так как отсутствует какое-либо сравнение: моделей между собой или модели на разных промежутках времени. Сигнал характеризует морфологию среды для маркировки состояний или действий важных для динамики этой конкретной среды. Например, метод расширения возможностей [357; 642] определяет информационную емкость между последовательностью действий A^h и конечным состоянием \tilde{S} , что характеризует возможности контроля агентом среды из текущего состояния S . Еще одним примером является сигнал достижимости, позволяющий задавать внутренние цели агенту [253; 643], которые можно достичь за определенное количество действий.

Среди методов, основанных на морфологии среды, существует набор алгоритмов, которые не используют сигнал вознаграждения явно, но используют способ построения модели (в табл. 7 обозначены $\chi[M]^*$) как реализацию идеи внутренней мотивации.

Сигнал вознаграждения напрямую оказывает влияние на стратегию агента. Так, внешнее вознаграждение определяет основное — целевое — поведение агента. Перед исследователями стоит тяжелая задача определения такого сигнала, который не позволяет алгоритму стагнировать в локальных оптимумах и обеспечивает обучение за приемлемое время. Тесно связана с этим проблема разреженности сигнала вознаграждения (когда агент не может продвинуться в обучении, так как не получает сигнала обратной связи). Один из способов преодоления этой трудности — добавление к редкому внешнему сигналу дополнительного плотного сигнала внутреннего вознаграждения (классический пример: решение задачи прохождения игры «Месть Монтесумы» [255]).

Смешивание внешнего и внутреннего вознаграждений определяется линейной комбинацией:

$$[R, R_{int}] : r = R + \alpha R_{int}, \quad (3.12)$$

где r уже используется в классическом варианте обучения стратегии агента 3.6. Например, такой подход реализован в работах [201; 246; 637]. Помимо смешивания вознаграждений, возможно смешивание функций полезности (например, алгоритм MEEE [543]), что позволяет сохранить информацию о внешнем вознаграждении в функции полезности отдельно от исследовательского сигнала методов внутренней мотивации.

Смешанный сигнал вознаграждения вносит смещение в решение поставленного МППР. Для устранения этого эффекта необходимо подбирать адаптирующийся коэффициент α 3.12, который со временем затухает, либо сигнал внутреннего вознаграждения должен по мере обучения агента сходиться к нулю, что характерно для сигналов $L[M]$ и ΔM . Впрочем, есть исключения: некоторые варианты ошибки предсказания в случае проблемы шумного телевизора [201] будут выдавать постоянный ненулевой сигнал на неконтролируемый шум в среде.

Обучение внутренне мотивированного агента с применением модели мира для модификации вознаграждения агента дополнительным сигналом состоит из следующих этапов (см. рисунок 3.26). Обученная модель агента генерирует внутреннее вознаграждение. Сигнал внутренней мотивации, смешиваясь с внешним, обучает стратегию и обеспечивает её исследовательской компонентой. Такая стратегия позволяет скорректировать взаимодействие со средой для улучшения данных в памяти D_M 3.5. Модель вносит вклад в процесс, пока она не станет достаточно хорошо обученной (так что сигнал внутренней мотивации упадет до нуля) либо пока стратегия не станет успешной (коэффициент α должен в таком случае стать равным нулю).

Подмешивание внутреннего сигнала к внешнему вознаграждению связывает обучение стратегии агента с обучением модели мира, что вызывает некоторые проблемы. Обученная модель получается смещенной в контексте достижения одной цели, из-за чего для эффективного решения новой задачи агенту требуется дополнительное обучение для настройки под задачу. Обозначенная проблема нивелируется введением дополнительной стратегии исследования (см. рисунок 3.26), которая не связана с целевой стратегией, так как единственной задачей исследовательской стратегии является сбор наблюдений, не зависящих от поставленной перед агентом внешней цели.

Обучение внутренне мотивированного агента с применением исследовательской стратегии состоит из обучения исследовательской стратегии на внутреннем вознаграждении по наблюдениям из среды или из модели (см. рисунок 3.26). Исследовательская стратегия, полностью определяющая управление агентом, собирает данные только в среде, обучая модель агента, которая впоследствии используется для нахождения решения конкретных задач. Так, агент, имея модель мира, без взаимодействия со средой способен научиться достигать цель, поставленную внешней мотивацией, и сразу на приемлемом уровне выполнить ее в среде. Например, в работах [499; 566] исследовательские стратегии (Plan2Explore и MAX соответственно) набирают выборку D_M , опираясь на дисперсию ансамбля моделей, а обучение самой стратегии происходит по модели в «воображении» (т.е. выборка D_ϵ определяется моделью M). Похожей схеме работает агент CEE-US [548]; однако он использует ансамбль графовых сетей. В обучении модели, т.е. сборе выборки D_M , может также принимать участие и целевая стратегия, что вносит необходимость определения соотношения между данными от π_ϵ и π_g , где доля примеров разных стратегий является гиперпараметром агента (например, см. [227]).

Помимо обучения модели для решения неизвестных ранее заданий, исследовательская стратегия используется в качестве инициализации для целевой стратегии. Так, например, в работе [335] авторы предлагают сохранять копии исследовательской стратегии по ходу обучения, чтобы затем использовать их в качестве набора навыков, которые, например, можно применять в задачах иерархического обучения с подкреплением. Еще одним применением исследовательской стратегии является определение внутренне мотивированных целей (например, агент LEXA [227]).

Для обучения агента достижению целей необходимо определить пространство целей и задать порядок их выбора. Стандартным способом определения пространства целей является

отождествление его с пространством состояний. Выбор целей сводится к модифицированной задаче МППР: агент представляет иерархическую структуру, где на нижнем уровне стратегия $\pi(a|s,g)$ определяет действия, а на следующем уровне стратегия $\pi(g|s)$ выбирает цели g для нижнего уровня. Такой подход позволяет использовать остальные методы внутренней мотивации для определения стратегии верхнего уровня. Например, Director [212] использует внутреннее вознаграждение для корректировки стандартного вознаграждения в стратегии «менеджера» $\pi(g|s)$, определяющего цели для стратегии «работника» $\pi(a|s,g)$.

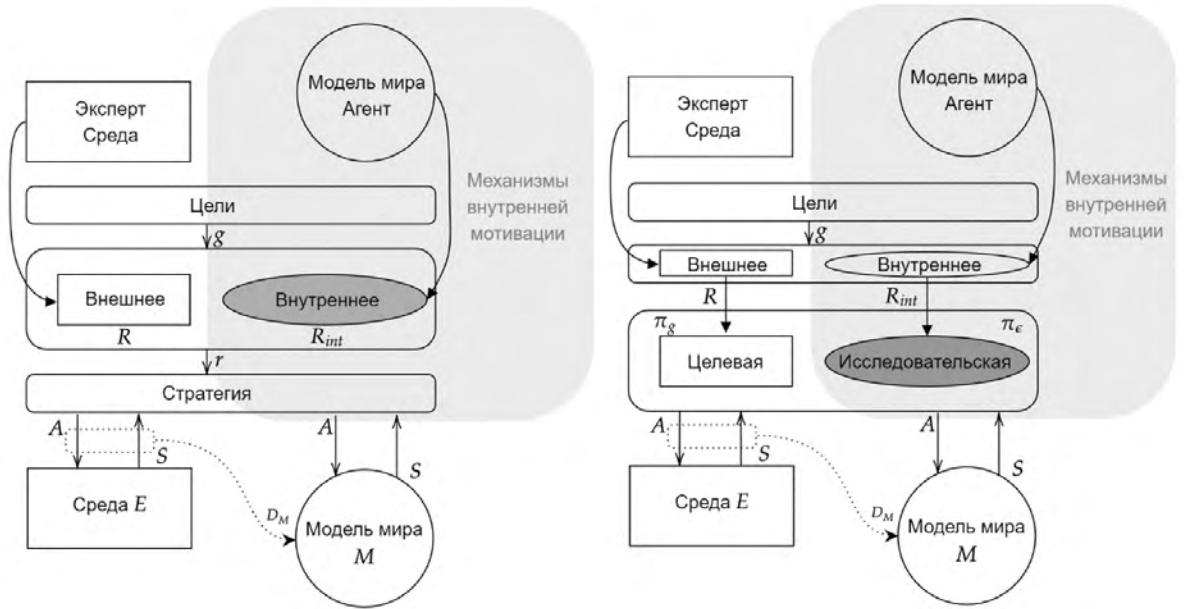


Рисунок 3.26 — Слева: методы внутренней мотивации явным образом модифицируют вознаграждение агента. Справа: методы внутренней мотивации реализуют исследовательскую стратегию, явно корректируя стратегию агента.

Многие агенты не используют такое однородное иерархическое представление, а определяют множество целей из истории взаимодействия агента со средой или моделью мира. В этом случае алгоритм обучения стратегии разбивается на две подзадачи: набрать множество целей или сконструировать пространство целей; выбрать цели из сформированного ранее набора.

Формирование множества целей. Модель мира позволяет определять набор целей, которые улучшают её (см. рисунок 3.27). Целями становятся состояния, в которых модель плохо делает предсказания. Подобные цели направляют агента набрать больше опыта для мало изученных состояний. Например, алгоритм LEXA [227] использует исследовательскую стратегию для того, чтобы найти новые, плохо изученные состояния, которые становятся целями для параллельной стратегии достижения целей.

Модель мира содержит информацию о достижимости тех или иных состояний, что позволяет выделить среди всего их множества те, которые агент может достичь. В этом подходе обычно используется упрощенная версия модели, которая определяет вероятность достичь одного состояния из другого за некое фиксированное число шагов (например, [643]).

Еще один вариант — когда в представление целей закладывается связь с состояниями, из которых она достижима (например, CC-RIG [194]).

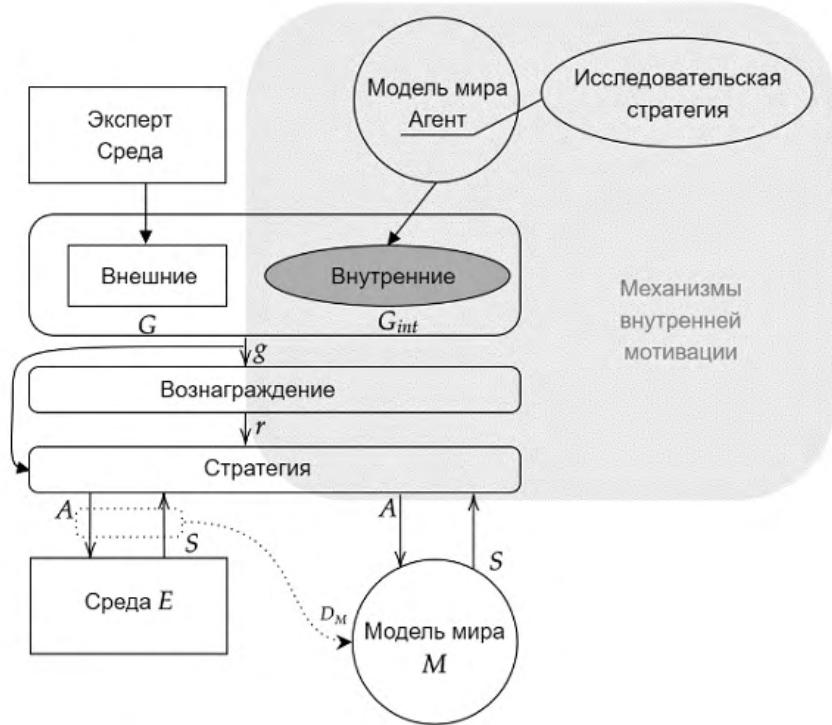


Рисунок 3.27 — Модель мира явно определяет цели исходя из морфологии среды и неявно через определение исследовательской стратегии, достигнутые которой состояния становятся целями.

Построение модели мира в определенной форме позволяет ввести удобное в алгоритмическом смысле представление переходов и состояний в среде, выделяющее морфологию мира, из которой явно определяется набор целей (соответствует $\chi[M]^*$ сигналу мотивации). Например, в алгоритме SMORL [655] используется объектное, факторизованное представление состояний: на основе сырой сенсорной информации формируется вектор, компоненты которого соответствуют характеристикам отдельных объектов. В другом методе (алгоритм SRICS [654]) модель определяет граф взаимодействий (вершины — отдельные объекты), который позволяет разложить достижение общей цели на подцели, образующие множество внутренних целей для агента.

Выбор целей без стратегии верхнего уровня во многом определяется по тем же самым принципами, согласно которым формировалось пространство целей. При формировании множества целей используется численный критерий отбора (например, вероятность достижимости, см. выше), который определяет приоритет выбора цели. Другим способом выбора последовательности обучения целей является использование графа взаимодействий, описывающего модель мира. Например, в алгоритме SRICS [654] граф задает последовательность обучения отдельных вершин (целей).

3.5 Выводы

В данной главе диссертационного исследования были рассмотрены различные реализации модуля одновременного обучения и планирования в архитектуре NSLP. Универсальным семейством алгоритмов обучения с подкреплением для реализации этого модуля является семейство «актор-критик». В данной главе были рассмотрены различные варианты интеграции модели среды для осуществления планирования и ускорения обучения субоптимальной стратегии агента. В первом разделе был рассмотрен вариант интеграции модели среды на основе функции полезности в часть критика. Данный подход показал многообещающие результаты на визуальных средах управления поведением агента, таких как Atarai и Crafter.

Во втором разделе был рассмотрен вариант реализации интерпретируемой модели среды для предсказания изменений объекта управления в непрерывном пространстве действий. Применяемая техника нейросетевых обыкновенных дифференциальных уравнений позволяет генерировать достаточно точные траектории, чтобы пополнять память прецедентов для «актора-критика» и таким образом ускорять обучения по сравнению с безмодельными алгоритмами и алгоритмами типа Dreamer, не специализированными для задач робототехнического управления.

В третьем разделе были рассмотрены многоагентные постановки задачи генерации стратегии поведения агента на примере решения задачи поиска пути. Были разработаны два алгоритма, MAMCTS и MATS-LP, которые используют готовое графовое объектное представление среды для моделирования ее динамики в виде построения дерева Монте-Карло. Были представлены результаты решения задач MAPF и LMAPF на различных картах, превосходящие все современные алгоритмы. Наконец, в заключении была рассмотрена проблема исследования среды при одновременном обучении стратегии агента и модели среды. Были разработаны подход и общая методология включения обучения по внутреннему вознаграждению в архитектуру NSLP.

Глава 4. Объектно-центричные представления в архитектуре NSLP

В данной главе предложена объектно-центричная парадигма в обучении с подкреплением на основе модели. Отдельное внимание уделено объектно-центричным моделям мира, в которых в явном виде используются интерпретируемые представления отдельных объектов на сцене и динамика среды моделируется на основе предсказания взаимодействия этих объектов друг с другом. Основные результаты, представленные в данной главе, были опубликованы в [16] (раздел 4.2), [17; 72] (раздел 4.1), [55; 58] (раздел 4.3), [8] (раздел 4.4).

4.1 Распутанные объектно-центричные представления сцен

В данном разделе диссертационного исследования продолжено рассмотрение нейросимвольного уровня работы архитектуры NSLP (рисунок 4.4), но уже не в слотовой реализации, а с использованием распутанных латентных нейросетевых представлений. Также исследованы особенности организации работы нейросимвольной объектной модели $NS(o_t)$, которая на входе принимает исходное или сжатое представление о наблюдении агента, на выходе генерируя набор представлений объектов в интерпретируемом распутанном виде. Основные результаты, изложенные в данном разделе, представлены в публикациях [17; 72].

Как уже отмечалось в предыдущем разделе, хорошо известная проблема нейронных сетей заключается в том, что они не могут проводить обобщение сенсорной информации на уровне, сопоставимом с человеческим [298]. Считается, что причиной этому является проблема привязки символов, то есть неспособность современных нейросетевых моделей динамически и гибко связывать распределенную в них информацию в некоторую абстракцию. Эта проблема влияет на способность нейронных сетей

- создавать значимые представления абстракций (сущностей) из неструктурированных сенсорных входных данных;
- поддерживать полученное разделение информации на уровне представления этих сущностей;
- повторно использовать эти представления для формирования новых выводов и предсказаний.

Один из способов решить эту проблему — ограничить нейросетевую модель формированием распутанных объектно-центричных представлений сцены [444; 453].

Распутанное объектно-центричное представление (disentangled object-centric representation) потенциально может улучшить обобщение моделей и интерпретируемость получаемых результатов во многих областях машинного обучения, таких как

структурированное представление и генерация сцен [628], обучение с подкреплением [184; 506], моделирование рассуждений [469] и объектно-центричные визуальные задачи [562; 573]. Однако, за немногими исключениями [406; 444; 453], недавние исследования были сосредоточены либо на объектно-центричных, либо на распутанных представлениях и не уделяли достаточного внимания вопросу их объединения.

В данном разделе предлагается модель, которая формирует распутанное представление объектов путем квантования соответствующего слотового представления. Данный подход называется векторно-квантованным слотовым вниманием (VQ-SA). Квантование слотов, полученных в результате обучения без учителя [471], включает в себя два этапа. Сначала инициализируются дискретные скрытые пространства, каждое из которых соответствует одному из порождающих факторов в данных. Во-вторых, каждое скрытое пространство инициализируется векторными представлениями по количеству соответствующих значений порождающего фактора. Это двухэтапное квантование позволяет модели присвоить конкретному векторному представлению конкретное значение порождающего фактора. Предлагаемое распутанное объектно-центричное представление улучшает результаты базовой модели [471] для стандартных объектно-центричных визуальных задач, таких как прогнозирование множества свойств, даже по сравнению со специализированными для этой задачи моделями [659]. Данный вывод подтверждается с помощью обширных экспериментов на наборе данных CLEVR [179].

Обычно используемые показатели качества распутывания [154; 350] основаны на предположении, что распутывание достигается на уровне координат вектора, т.е. каждая координата соответствует порождающему фактору. В предлагаемом подходе порождающие факторы выражаются векторами, а отдельные координаты не поддаются интерпретации. Таким образом, такие метрики не подходят, и проблема количественной оценки распутывания остается открытым вопросом. Тем не менее, были предложены метрики DQCF-микро и DQCF-макро, которые качественно оценивают запутанность в задаче обнаружения объекта. Оригинальная модель слотового внимания [471] достигает отличных результатов в задаче обнаружения объектов, но предлагаемая модель позволяет разделять распределенные признаки не только по представлениям объектов, но и по представлениям их значений.

Основной вклад данного раздела диссертационного исследования в области объектно-центричных представлений заключается в следующем:

1. была предложена дискретная реализация объектно-центричных векторных представлений, которая отображает их в отдельные скрытые пространства;
2. процедура квантования создает распутанное представление, где распутывание достигается на уровне скрытых представлений;
3. обученные дискретные представления позволяют манипулировать отдельными объектами на сцене, изменяя представления в скрытом пространстве;
4. модель VQ-SA обеспечивает наилучшие результаты в задаче прогнозирования множества свойств на наборе данных CLEVR среди класса современных объектно-центричных методов;

5. были предложены метрики DQCF-микро и DQCF-макро, которые качественно оценивают распутывание полученных дискретных переменных, представленных векторами.

4.1.1 Квантованные объектно-центричные представления

Квантованные распутанные представления объектов получаются в результате двухстадийного процесса. Сначала обнаруживаются объекты с помощью слотового внимания (Slot Attention) [471], а затем они преобразуются их необходимые квантованные представления. Идея слотового представления состоит в том, чтобы сопоставить входные данные (изображение) с набором скрытых переменных (слотов) вместо сопоставления с одним скрытым вектором [351; 531]. Это происходит таким образом, чтобы каждый слот описывал часть входных данных [252; 278; 471]. Каждому объекту назначается слот размерности d_s , и каждый слот преобразуется в необходимое скрытое представление. Такой подход инспирирован дискретными скрытыми представлениями, предложенными в работе [479]. Вместо использования одного дискретного скрытого пространства для отображения слотов в данном подходе предлагается использовать несколько скрытых пространств с небольшим размером векторных представлений d_l ($d_l < d_s$) и небольшим количеством самих представлений в каждом скрытом пространстве.

Основное предположение, стоящее за предложенной двухстадийной процедурой, заключается в том, что каждый объект генерируется фиксированным числом порождающих факторов. Таким образом, каждый объект можно закодировать как комбинацию векторных представлений, соответствующих значениям порождающих факторов. Для используемых данных большинство порождающих факторов являются дискретными, и подходящим выбором порождающего распределения было бы категориальное. Выбранное количество категориальных распределений равно количеству порождающих факторов, при этом количество возможных категорий равно количеству значений факторов. Общая архитектура предлагаемой модели VQ-SA показана на рисунке 4.1.

Процесс работы метода VQ-SA для задачи прогнозирования множества свойств состоит из трех этапов. На этапе предварительной обработки (рисунок 4.1а) изображение I переводится в векторное представление с помощью кодировщика в сочетании с позиционными представлениями. Объекты распределяются по слотам с помощью механизма слотового внимания. На втором этапе (рисунок 4.1б) выполняется раздельная обработка категориальных порождающих факторов (CatGF) и непрерывных порождающих факторов (ConGF). На этой стадии используется квантование слотов по множеству дискретных скрытых пространств для представления категориальных порождающих факторов и оценки значений непрерывных порождающих факторов. На третьем шаге (рисунок 4.1с) оба типа порождающих факторов объединяются, и прогнозируются свойства объекта. Предсказанные объекты сопоставляются с истинными представлениями, используя

венгерский алгоритм [368]. Для задачи обнаружения объектов непрерывные порождающие факторы обрабатываются аналогично порождающим факторам (рисунок 4.1b) и используется та же стратегия декодирования, что и в алгоритме слотового внимания (рисунок 4.1d).

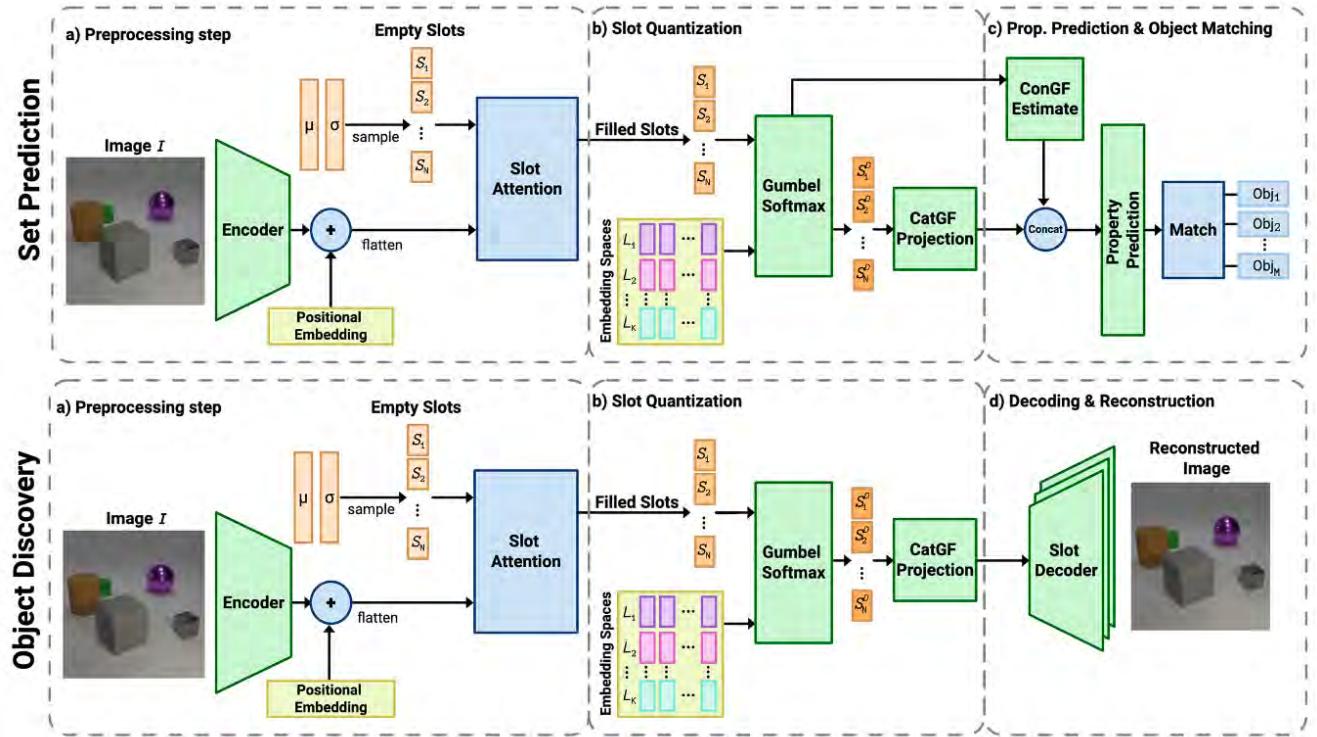


Рисунок 4.1 — Общая архитектура предлагаемой модели VQ-SA.

Чтобы представить изображение I в виде набора слотов $S_i, i = 1 \dots N$, где слоты являются векторами размерности d_s , предлагается следовать стандартной процедуре, предложенной в работе [471]. Каждый слот S_i инициализируется случайным образом $S_i = \mu + \sigma \cdot N(0,1)$ и используется для итеративного сопоставления с объектом $O_m, m = 1 \dots M$ или фоном изображения I . Основной особенностью механизма слотового внимания является то, что слоты конкурируют друг с другом за сопоставление с объектом, т.е. коэффициенты внимания нормализуются по слотам.

После назначения объектов слотам каждый слот S_i представляется в дискретных скрытых пространствах $L_k, k = 1 \dots K$, где K — это количество скрытых пространств, соответствующее количеству категориальных порождающих факторов ($K = 4$ для набора данных CLEVR). Каждое скрытое пространство L_k инициализируется векторными представлениями $e_j^k, j = 1 \dots n_k$ размерностью d_l ($d_l < d_s$). Здесь n_k — это число векторных представлений в скрытом пространстве L_k , т.е. число категорий порождающего фактора. Затем слот S_i линейно проецируется в более низкую размерность d_l : $S'i = MS_i$ ($S'i \in \mathbb{R}^{d_l \times 1}$, $M \in \mathbb{R}^{d_l \times d_s}$, $S_i \in \mathbb{R}^{d_s \times 1}$). После этого создается новое представление слота S' в каждом дискретном скрытом пространстве L_i с помощью процедуры мягкого максимума Гюмпеля [336]. Сначала вычисляется сходство между слотом $S'i$ и каждым векторным представлением e_j^k в скрытом пространстве L_k , чтобы получить апостериорные

распределения $q(e_j^k | S'i)$ путем нормализации с помощью функции мягкого максимума:

$$sim^k = (S'_i)^T L_k = (S'_i)^T [e_1^k, e_2^k, \dots, e_{n_k}^k], \quad q(e_j^k | S'_i) = \frac{\exp(sim_j^k)}{\sum_j \exp(sim_j^k)}. \quad (4.1)$$

Чтобы получить непрерывную аппроксимацию y^k унитарно (one-hot) закодированного представления дискретной переменной e^k , также используется процедура мягкого максимума Гюмпеля, но с постоянным температурным параметром $t = 2$:

$$y_j^k = \frac{\exp\left(\frac{(g_j + \log(q(e_j^k | S'_i)))}{t}\right)}{\sum_j \exp\left(\frac{(g_j + \log(q(e_j^k | S'_i)))}{t}\right)}, \quad (4.2)$$

где g обозначает случайные выборки из распределения Гюмпеля.

Результирующее представление \hat{e}_i^k слота $S'i$ в дискретном скрытом пространстве L_k представляет собой взвешенную сумму всех векторных представлений $e_j^k \in L_k$ из L_k с весами $y^k = [y_1^k, y_2^k, \dots, y_{n_k}^k]$: $\hat{e}_i^k = (y^k)^T L_k = [y_1^k, y_2^k, \dots, y_{n_k}^k]^T [e_1^k, e_2^k, \dots, e_{n_k}^k]$.

Затем представления $S'i$ из всех дискретных скрытых пространств $L_k, k = 1 \dots K$ объединяются: $S_i^D = [\hat{e}_i^1, \dots, \hat{e}_i^K]$. S_i^D далее используется в качестве квантованного представления слота $S'i$. Данное объединение можно рассматривать как процесс построения нового векторного представления S_i^D из отдельных порождающих факторов $\hat{e}_i^1, \dots, \hat{e}_i^K$.

Кодирование непрерывных порождающих факторов, таких как координаты пространственного положения объектов, дискретными скрытыми пространствами может привести к значительным ошибкам квантования. Кроме того, не существует универсального способа представления непрерывного значения дискретным интервалом, в связи с чем длина интервала может рассматриваться как гиперпараметр.

В задаче прогнозирования множества свойств явно проводится разделение порождающих факторов на категориальные (форма, цвет и т.д.) и непрерывные (координаты x, y, z). В данном подходе используется квантование слота для представления категориальных порождающих факторов, и получается квантованное представление слота S_i^D . Для прогнозирования же координат объекта используется двухслойный персептрон (MLP). В задаче обнаружения объекта координаты также обрабатываются отдельно, но способом, аналогичным категориальным порождающим факторам. Единственное отличие заключается в том, что в этом случае используется одно скрытое пространство вместо нескольких скрытых пространств.

Чтобы облегчить распутывание дискретных скрытых пространств, в предлагаемом подходе добавляется следующий хорошо известное слагаемое к стандартной функции потерь: $-\sum_k KL(q(L_k | S') \| p(L_k))$, где $p(L_k)$ — истинное априорное равномерное категориальное распределение по дискретному скрытому пространству L_k , $q(L_k | S')$ — апостериорное категориальное распределение по дискретному скрытому пространству, предсказанное нейросетевой моделью. Используя этот член функции потерь, удается добиться того, что апостериорные распределения по каждому скрытому пространству $q(L_k | S')$ оказываются независимыми и более близкими к априорным распределениям, что приводит к лучшему разделению пространств.

4.1.2 Экспериментальное исследование квантованных объектно-центричных представлений

Для всех рассматриваемых в данном разделе задач сначала была обучена исходная модель слотового внимания на 600 тысячах итераций, а затем его кодировщик веса был использован для инициализации соответствующих модулей VQ-SA. Был проведен также эксперимент со сквозным обучением, но модель в этом случае сходится существенно медленнее, что затрудняет проведение достаточного количества экспериментов.

Визуальные объектно-центричные задачи

В задаче прогнозирования множества свойств модель получает изображение в качестве входных данных и прогнозирует целевые признаки в виде неупорядоченного набора объектных векторов. Векторы прогнозируемых и целевых признаков сопоставляются с использованием венгерского алгоритма [368]. Для количественной оценки качества модели используется показатель средней точности (AP_{thr}) с некоторым пороговым значением. Обнаруженный объект считается истинным положительным примером, если:

1. набор его предсказанных признаков точно соответствует истинным значений и
2. местоположение объекта предсказано в пределах порогового значения thr относительно истинных непрерывных значений.

Порог ∞ означает, что пороговое расстояние не используется.

Как показано в табл. 8, VQ-SA значительно улучшает результаты в этой задаче для малых пороговых значений: более чем в два раза для порога 0,25 и более чем в 2,5 раза для порога 0,125. По сравнению с узкоспециализированной моделью для задачи прогнозирования множества свойств, iDSPN [454], VQ-SA показывает сопоставимые результаты для пороговых значений, превышающих единицу, и значимые результаты для меньших пороговых значений. Причина таких результатов заключается в том, что архитектура предлагаемой модели не была специально настроена для задачи прогнозирования множества свойств и может быть использована для других задач, таких как обнаружение объектов.

Чтобы обеспечить корректное сравнение с моделью слотового внимания, были использованы те же гиперпараметры во время обучения: размер пакета данных 512, три итерации слотового внимания и 150000 итераций обучения. Модель обучается с использованием оптимизатора Adam [352] со скоростью обучения 0,0004. Был также использован так называемый «прогрев» скорости обучения с экспоненциальным затуханием после этого. Количество слотов равно 10, поскольку используется набор данных CLEVR [179], где представлены изображения с десятью объектами или менее.

Таблица 8 — Результаты работы на задаче прогнозирования множества свойств объектов на наборе данных CLEVR.

Model	AP_{∞} (%)	AP_1 (%)	$AP_{0.5}$ (%)	$AP_{0.25}$ (%)	$AP_{0.125}$ (%)
Slot MLP	19.8 ± 1.6	1.4 ± 0.3	0.3 ± 0.2	0.0 ± 0.0	0.0 ± 0.0
DSPN T=30	85.2 ± 4.8	81.1 ± 5.2	47.4 ± 17.6	10.8 ± 9.0	0.6 ± 0.7
DSPN T=10	72.8 ± 2.3	59.2 ± 2.8	39.0 ± 4.4	12.4 ± 2.5	1.3 ± 0.4
Slot Attention	94.3 ± 1.1	86.7 ± 1.4	56.0 ± 3.6	10.8 ± 1.7	0.9 ± 0.2
VQ-SA (ours)	96.1 ± 0.4	91.2 ± 0.5	71.8 ± 2.3	22.2 ± 2.1	2.4 ± 0.2
iDSPN	98.8 ± 0.5	98.5 ± 0.6	98.2 ± 0.6	95.8 ± 0.7	76.9 ± 2.5

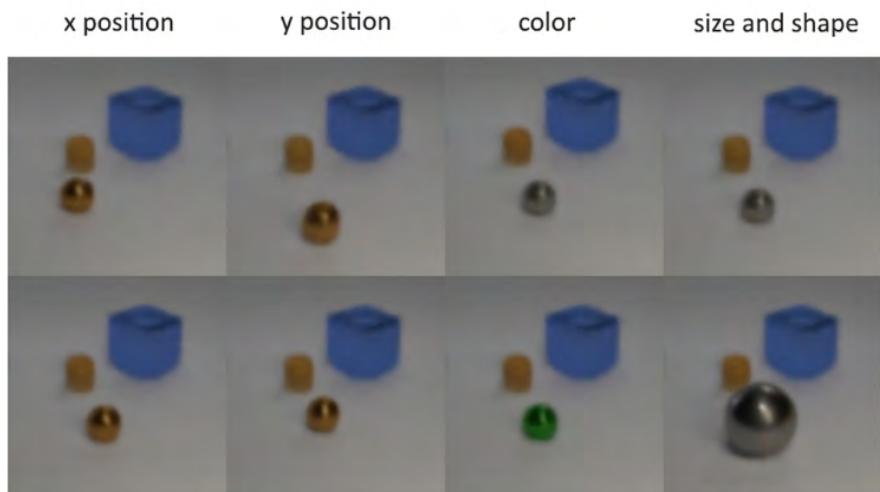


Рисунок 4.2 — Манипулирование отдельными объектами в сцене.

В задаче обнаружения объектов модель получает на входе изображение, разделяет распределенные признаки сцены на отдельные признаки объектов и использует их для реконструкции объектов, объединяя их затем в исходное изображение сцены. Оригинальный алгоритм слотового внимания достигает отличных результатов в задаче обнаружения объектов, в то время как предлагаемое расширение позволяет модели разделять распределенные признаки не только по представлениям объектов, но и по представлениям их свойств. Рисунок 4.2 иллюстрирует возможность манипулирования отдельными объектами в сцене с помощью разрабатываемой модели: столбцы соответствуют изменению дискретного компонента представления объекта в дискретную скрытую переменную из того же пространства.

Здесь использованы те же настройки процесса обучения, что и в исходном алгоритме слотового внимания: основным компонентом функции потерь является среднеквадратичная ошибка восстановления изображения, размер пакета данных равен 64, а настройки оптимизатора и скорости обучения такие же, как и в задаче прогнозирования множества свойств. Здесь также используется дополнительное слагаемое в функции потерь, который помогает сформировать распутанные представления.

В исследованиях влияния отдельных компонент модели (см. табл. 9) были продемонстрированы эффект увеличения числа векторных представлений в каждом

пространстве (до 8 и 16) и эффект использования только одного общего пространства (cspace) в задаче прогнозирования множества свойств. Данные модификации не сказываются на общей производительности.

Таблица 9 — Результаты исследования влияния компонент модели. В скобках указано общее количество векторных представлений.

Model	AP _∞ (%)	AP ₁ (%)	AP _{0.5} (%)	AP _{0.25} (%)	AP _{0.125} (%)
Slot Attention	94.3 ± 1.1	86.7 ± 1.4	56.0 ± 3.6	10.8 ± 1.7	0.9 ± 0.2
VQ-SA (ours)	96.1 ± 0.4	91.2 ± 0.5	71.8 ± 2.3	22.2 ± 2.1	2.4 ± 0.2
VQ-SA(8, cspace)	41.8 ± 27.5	38.2 ± 26.5	27.6 ± 18.3	7.0 ± 4.4	0.7 ± 0.4
VQ-SA(16, cspace)	83.3 ± 7.7	79.3 ± 7.2	58.8 ± 4.5	15.3 ± 1.2	1.5 ± 0.1
VQ-SA(32, cspace)	92.3 ± 1.4	88.1 ± 1.4	66.0 ± 2.4	17.7 ± 1.6	1.8 ± 0.2
VQ-SA(64, cspace)	94.2 ± 0.6	90.3 ± 0.5	69.2 ± 2.6	19.1 ± 2.2	1.9 ± 0.3
VQ-SA(32)	95.9 ± 0.2	92.1 ± 0.4	70.4 ± 1.6	18.9 ± 1.4	1.9 ± 0.2
VQ-SA(64)	96.1 ± 0.1	92.1 ± 0.3	69.6 ± 1.1	18.5 ± 1.0	1.8 ± 0.1

Качество распутанных представлений

Чтобы качественно оценить распутывание получаемых в процессе обучения дискретных переменных, в данном разделе предлагаются метрики DQCF-micro и DQCF-macro (DQCF, Disentanglement Quality of Categorical generative Factors — качество распутывания категориальных порождающих факторов). DQCF-micro оценивает распутывание по отношению ко всем другим векторам из всех дискретных пространств, в то время как DQCF-macro оценивает его на уровне дискретных пространств. Для метрики DQCF-micro для каждого набора объектов из валидационной выборки вычисляется частота выборки каждого скрытого вектора как наиболее похожего. Эта статистика с венгерским сопоставлением дает частотные вероятности каждого свойства со значениями, обусловленными выбранными скрытыми векторами. Результаты для первого скрытого пространства показаны на рисунке 4.3(слева). Высокая близость распределения значений свойств к равномерному распределению означает, что этот вектор не является специфичным для объектов с определенным значением свойства и не содержит информации, уникальной для этого свойства.

Для метрики DQCF-macro вычисляется стандартное отклонение по всем значениям соответствующих свойств и получается среднее значение по всем векторам из одного и того же пространства. Результаты показаны на рисунке 4.3(справа). Полученные значения показывают, насколько изменение свойств объекта влияет на изменение распределения по данному пространству. Высокие значения указывают на то, что скрытые переменные из этого пространства содержат информацию, специфичную для этого свойства. Можно видеть,

что информация, специфичная, например, для свойства «цвет», содержится в векторах как третьего, так и четвертого пространства, т.е. они запутаны, в то время как информация о других свойствах распределена по векторам отдельных пространств.

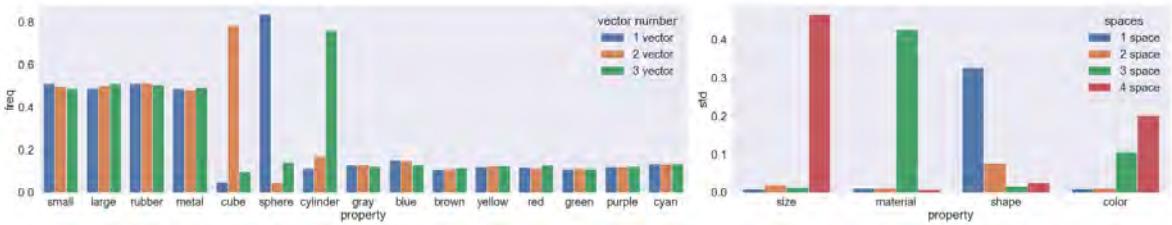


Рисунок 4.3 — Слева: результаты подсчета метрики DQCF-micro. Справа: результаты подсчета метрики DQCF-macro.

В данном разделе диссертационного исследования была предложена модель VQ-SA, которая получает распутанное представление объектов путем квантования соответствующего слотового представления. Описанный подход позволяет достичь самых высоких результатов среди класса объектно-центрических методов в задаче прогнозирования множества свойств. Было также показано, что, манипулируя обученными дискретными представлениями, удается генерировать объекты сцены с заданным свойством. Для качественной оценки степени запутанности представлений в задаче обнаружения объектов были предложены метрики DQCF-micro и DQCF-macro.

Важной особенностью предложенной модели является то, что в ней используется количество скрытых дискретных пространств, равное количеству порождающих факторов в данных. Следовательно, эта модель не может быть непосредственно применена к данным с другим количеством порождающих факторов. Как и в случае с большинством объектно-центрических моделей, были продемонстрированы результаты на относительно простых наборах синтетических данных. Модификация предложенной модели для более сложных сцен, реальных или так же синтетических, является многообещающей и сложной задачей для дальнейших исследований. В следующем разделе диссертационного исследования при рассмотрении уже собственно модификации алгоритма слотового представления показано, как можно улучшить показатели качества работы объектно-центрических алгоритмов на более реалистичных наборах данных.

4.2 Слотовое объектно-центрическое представление сцен

В данном разделе диссертационного исследования рассмотрена слотовая реализация нейросимвольного уровня работы архитектуры NSLP (рисунок 4.4). Исследованы особенности организации работы нейросимвольной объектной модели $NS(o_t)$, которая на входе принимает исходное или сжатое представление о наблюдении агента, на выходе генерируя набор представлений объектов в слотовом виде. Основные результаты, изложенные в данном разделе, представлены в публикациях [16].

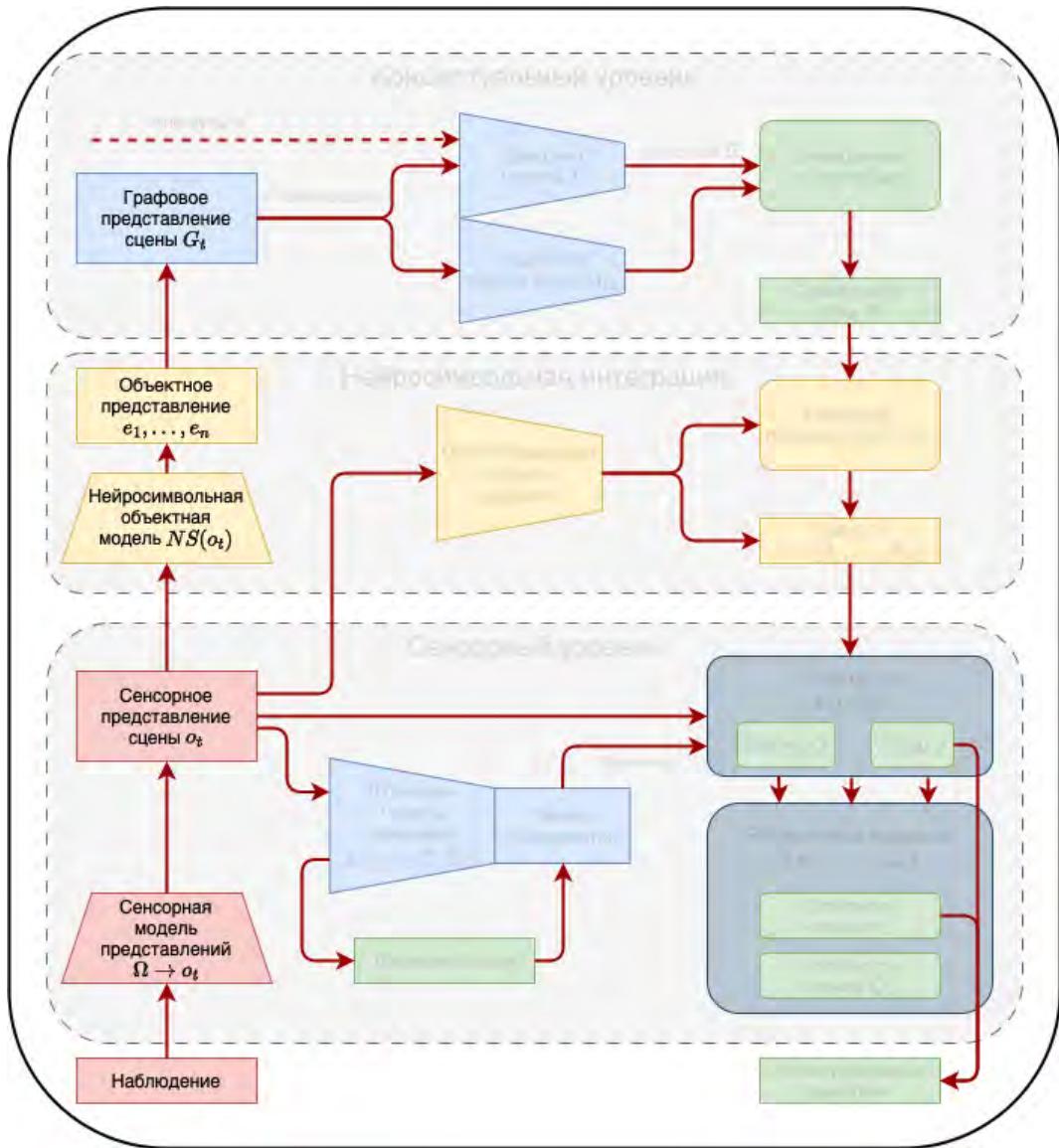


Рисунок 4.4 — Архитектура NSLP с выделенным нейросимвольным уровнем, на котором используется слотовый подход для построения объектного представления наблюдений агента.

4.2.1 Объектно-центричные представления сцен

Как уже отмечалось в разделе 4.1, в последние годы значительно возрос интерес к объектно-центрическим представлениям сцен [252; 278; 406; 444; 453; 471]. Такие представления потенциально могут улучшить способность многих методов машинного обучения к обобщению в различных областях, таких как обучение с подкреплением [184; 236; 258; 506; 631], представление и генерация сцен [607; 628; 631], моделирование рассуждений о визуальных сценах [469], специфичные объектно-центрические визуальные задачи [180; 562; 573] и, что наиболее важно для данного диссертационного исследования, алгоритмы визуального планирования [433].

Автоматическая сегментация объектов на сцене и формирование структурированного латентного пространства представлений могут быть реализованы различными способами [298]: дополнение выделяемых признаков информацией о принадлежности некоторой группе (объекту), использование представления тензорного произведения, привлечение идей из динамики атTRACTоров и т.д. Однако наиболее распространенным и эффективным методом получения объектно-ориентированных представлений является слотовый подход в реализации алгоритма слотового внимания (Slot Attention) [471]. Метод слотового внимания сопоставляет входной вектор признаков, полученный от сверточного кодировщика, с фиксированным числом выходных векторов признаков, называемых слотами. В процессе обучения каждому объекту присваивается соответствующий слот. Если количество слотов превышает количество объектов, то некоторые слоты остаются пустыми (т.е. они не содержат информации об объектах). Этот подход показал значительные результаты в объектно-центрических задачах, таких как прогнозирование множества свойств и обнаружение объектов (см. раздел 4.1).

Ограничения этой модели заключаются в том, что значимые результаты в задаче обнаружения объектов достигаются только на простых синтетических наборах данных, таких как CLEVR [179], Tetrominos и Multi-dSprites¹. На более сложных синтетических наборах данных, как, например, ClevrTex [348], или на реальных данных, таких как COCO [432], алгоритм слотового внимания показывает неудовлетворительные результаты. Кроме того, на датасете CLEVR, который является базовым набором данных для задачи прогнозирования множества свойств, слотовое внимание показывает значимые результаты только при низких значениях порога по точности непрерывных признаков, т.е. только в случаях, когда модель может сильно ошибаться относительно положения объектов. Другим недостатком слотового внимания является нестабильность его обучения, что затрудняет воспроизведение результатов.

Необходимо отметить, что в большинстве современных работ объектно-центрические методы оцениваются в основном только в задаче обнаружения объектов, т.е. при восстановлении масок этих объектов, что можно рассматривать как задачу обучения сегментации без учителя. В данном исследовании предполагается, что результаты задачи прогнозирования множества свойств также являются хорошим показателем качества для объектно-центрических методов, поскольку эта задача требует отличать объекты друг от друга и определять их принадлежность к заданному классу.

В данном разделе диссертационного исследования предложено обобщение алгоритма слотового внимания за счет замены метода k -средних на модель гауссовой смеси (Gaussian mixture model, GMM), которое называется модулем смеси слотов (slot mixture module, SMM). Такая замена приводит к более выразительным представлениям слотов за счет включения информации о расстоянии между кластерами и об отнесенных к ним векторах. Учет не только средневзвешенного значения множества векторов в качестве представления дает преимущество в разделении похожих, но все-таки различающихся групп. Например, два набора векторов, выбранных из разных распределений с одинаковым математическим

¹https://github.com/google-deepmind/multi_object_datasets

ожиданием, были бы неразличимы, если бы их среднее значение было единственным элементом представления. Этот метод улучшает объектно-центричное решение различных задач для разных наборов данных, превосходя даже узкоспециализированные модели в задаче прогнозирования множества свойств.

В данном разделе диссертационной работы были проведены тщательные экспериментальные исследования для обоснования работоспособности модели. С помощью широкого набора экспериментов было показано, что предлагаемый модуль смешивания слотов обеспечивает наилучшую производительность в задаче прогнозирования множества свойств в наборе данных CLEVR [179], превосходя в том числе узкоспециализированные модели, такие как [659]. Представлены экспериментальные результаты для задачи реконструкции изображений на четырех наборах данных: трех с синтетическими изображениями (CLEVR-Mirror [573], ShapeStacks [562], ClevrTex [348], Bitmoji²) и одном с реальными изображениями (COCO-2017 [432]). Предлагаемый модуль смешивания слотов последовательно улучшает производительность GPT-декодера. Показано, что модуль смешивания слотов превосходит исходный подход слотового внимания в задаче обнаружения объектов на таком сложном наборе данных как ClevrTex. В этом разделе также приводится пример использования предложенных улучшений слотового подхода для реализации идеи библиотеки концептов путем оценки параметров распределения этих концептов. Для обоснования замены k -средних на кластеризацию в стиле GMM было проведено сравнение подходов к кластеризации k -средних и GMM в задаче прогнозирования множества свойств и показано, что кластеризация GMM является лучшим выбором для объектно-центричного обучения.

Основные особенности предлагаемого подхода заключаются в следующем:

- предлагается обобщение подхода, основанного на слотах, для обучения объектно-центричных представлений, называемое модулем смешивания слотов;
- с помощью обширного набора экспериментов продемонстрировано, что предложенный подход последовательно повышает производительность в задачах обучения как без учителя (реконструкция изображения и обнаружение объектов), так и с учителем (прогнозирование множества свойств) с объектно-центричной постановкой и позволяет достичь самых высоких результатов в задаче прогнозирования множества свойств со строгими пороговыми значениями в наборе данных CLEVR;
- демонстрируется, как дополнительная оценка весов смеси слотов может помочь в построении качественной библиотеки концепций с отделением пустых и заполненных слотов. Предлагается также распространить этот подход на задачу выборки концептов.

²<https://www.kaggle.com/romaingraux/bitmojis/metadata/>

Слотовые объектно-центричные подходы для обучения векторных представлений

В процессе обучения без учителя объектно-центричное обучение предполагает идентификацию объектов на сцене, что является критичным элементом для формирования векторных представлений. Данная область развивалась, начиная с более ранних методов, которые использовали механизм последовательного внимания, такие как алгоритм AIR [136] и его развитие в работах SQAIR [560] и R-SQAIR [593].

В дальнейшем в этой области также появились методы, основанные на модели пространственных смесей, такие как Tagger [602] и NEM [297], которые направлены на оптимизацию полной вероятности модели смеси на уровне пикселей изображения. При этом последний подход фокусируется на обучении без учителя и группировке по перцептивным признакам. R-NEM [529] улучшил подход NEM, включив в него обнаружение взаимодействий объектов. Недавние работы продолжили интеграцию этих принципов, что привело к появлению более сложных подходов, таких как IODINE [453], MONET [444] и GENESIS [252; 278], которые используют метод вариационных автокодировщиков (VAE) [351; 531]. В то время как эти методы используют многоступенчатое кодирование и декодирование, гибридные модели, такие как SPACE [584], задействуют для декомпозиции сцены пространственное внимание.

Несмотря на разнообразие существующих подходов, общей характеристикой известных методов является переход к более эффективному представлению сложных сцен. Последовательные расширения для слотового внимания [192; 550] работают с видеоданными, в то время как в [334] было введено инвариантное слотовое внимание с целью повышения инвариантности представления объектов к геометрическим преобразованиям, а в [589] был интегрирован пространственно-локальный подход в объектно-центричные модели. Работа [161] представляет подход DINOSAUR, который использует самоконтролируемое обучение для дальнейшего повышения эффективности слотового внимания для работы в реальных сценариях обработки данных. Недавние достижения, такие как [467] и [202], хотя и не имеют прямого отношения к обучению представлений объектов, внесли значительный вклад в смежную область обучения сегментации объектов без учителя.

Нейросетевые модели, прогнозирующие множество свойств, применяются в различных задачах машинного обучения [248; 249; 392; 570]. Несмотря на свою практическую значимость, имеющиеся на данный момент модели часто испытывают сложности с представлением свойства множества. Однако такие подходы как глубокая сеть для прогнозирования множеств (Deep Set Prediction Network, DSPN) и ее улучшенная версия iDSPN [659] учитывают неупорядоченную природу множеств посредством ее моделирования за счет внутреннего цикла градиентного спуска и приближенного неявного дифференцирования, соответственно. Эквивариантные уровни самовнимания (self-attention)

также используются в таких моделях как TSPN для представления структуры множества свойств [363].

4.2.2 Модуль смеси слотов для объектного представления

Модуль слотового внимания

Модуль слотового внимания (SA) [471] реализует итеративную процедуру с использованием внимания, предназначенную для сопоставления распределенной карты признаков $\mathbf{x} \in \mathbb{R}^{N \times D}$ с набором K слотов $\mathbf{s} \in \mathbb{R}^{K \times D}$. Слоты инициализируются случайным образом, а обучаемая матрица $q \in \mathbb{R}^{D \times D}$ используется для получения проекций запросов к слотам, в то время как матрицы $k, v \in \mathbb{R}^{D \times D}$ используются для получения векторов ключей и значений по карте признаков \mathbf{x} . Внимание, вычисляемое по скалярному произведению [413] проекций q и k с помощью функции мягкого максимума (SoftMax) по q измерениям, подразумевает моделирование конкуренции между слотами за наибольший вклад в объяснение частей входных данных. Коэффициенты внимания $A \in \mathbb{R}^{N \times K}$ используются для присвоения v проекций слотам с помощью средневзвешенного значения:

$$M = \frac{1}{\sqrt{D}} k(\mathbf{x}) q(\mathbf{s})^T \in \mathbb{R}^{N \times K}, \quad A_{i,j} = \frac{e^{M_{i,j}}}{\sum_{j=1}^K e^{M_{i,j}}}, \quad (4.3)$$

$$W_{i,j} = \frac{A_{i,j}}{\sum_{i=1}^N A_{i,j}}, \quad \mathbf{s}^* = W^T v(\mathbf{x}) \in \mathbb{R}^{K \times D}. \quad (4.4)$$

Для уточнения значения слотов используется управляемый рекуррентный блок (gated recurrent unit, GRU) [390], принимающий на входе предварительно обновленные представления слотов \mathbf{s} в качестве скрытых состояний и обновленные слоты \mathbf{s}^* . Ключевой характеристикой слотового внимания является его инвариантность к перестановкам входных векторов и эквивариантность перестановок слотов. Это делает слотовое внимание подходящим методом для обработки множества свойств и построения объектно-центрических представлений.

Технически слотовое внимание является обучаемым аналогом алгоритма кластеризации k -средних с дополнительным обучаемым шагом обновления GRU и скалярным произведением (с обучаемыми проекциями q, k, v) вместо евклидова расстояния в качестве меры сходства между входными векторами и центроидами кластера. В свою очередь, кластеризацию k -средних можно рассматривать как частный случай модели гауссовой смеси.

Модель смеси распределений

Модели смесей (ММ) — это класс параметрических вероятностных моделей, в которых предполагается, что каждый \mathbf{x}_i из множества некоторых наблюдений $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \in \mathbb{R}^{N \times D}$ выбирается по распределению смеси с K компонентами смеси и априорными весами смеси $\boldsymbol{\pi} \in \mathbb{R}^K$:

$$\mathbf{x}_i \sim p(\mathbf{x}_i | \boldsymbol{\theta}) = \sum_{k=1}^K \pi_k p(\mathbf{x}_i | \boldsymbol{\theta}_k), \quad P(\mathbf{X} | \boldsymbol{\theta}) = \prod_{i=1}^N p(\mathbf{x}_i | \boldsymbol{\theta}), \quad \sum_k \pi_k = 1. \quad (4.5)$$

Такие модели можно рассматривать как модели со скрытыми переменными $z_{i,k} \in \{z_1, \dots, z_K\}$, которые указывают, из какого компонента был получен \mathbf{x}_i . Основная задача состоит в том, чтобы найти такие K групп параметров компонентов $\boldsymbol{\theta}_k$ и отнесение компонентов к каждой выборке \mathbf{x}_i , которые максимизируют правдоподобие модели $P(\mathbf{X} | \boldsymbol{\theta})$. Для решения этой задачи обычно используется алгоритм максимизации математического ожидания (ЕМ-алгоритм) — итеративный подход, включающий в себя два общих шага. Шаг **вычисления ожидания (Е)** реализует подсчет математического ожидания значения полной вероятности $P(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}^*)$ с учетом условного распределения $P(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})$:

$$Q(\boldsymbol{\theta}^*, \boldsymbol{\pi}^* | \boldsymbol{\theta}, \boldsymbol{\pi}) = \mathbb{E}_{P(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})} [\log P(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}^*)], \quad P(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}^*) = \prod_{i=1}^N \prod_{k=1}^K [\pi_k p(\mathbf{x}_i | \boldsymbol{\theta}_k^*)]^{I(z_i=z_k)}, \quad (4.6)$$

где $I(*)$ — индикаторная функция.

Шаг **Максимизации (М)** реализует процедуру поиска значений $\boldsymbol{\theta}^*, \boldsymbol{\pi}^*$, которые максимизируют значение $Q(\boldsymbol{\theta}^*, \boldsymbol{\pi}^* | \boldsymbol{\theta}, \boldsymbol{\pi})$:

$$(\boldsymbol{\theta}, \boldsymbol{\pi}) = \operatorname{argmax}_{(\boldsymbol{\theta}^*, \boldsymbol{\pi}^*)} Q(\boldsymbol{\theta}^*, \boldsymbol{\pi}^* | \boldsymbol{\theta}, \boldsymbol{\pi}). \quad (4.7)$$

Одной из наиболее широко используемых моделей такого рода является модель гауссовой смеси (GMM), где каждый компонент смеси моделируется как гауссово распределение, параметризованное его средними значениями и ковариационной матрицей, которая в простейшем случае является диагональной: $P(\mathbf{x}_i | \boldsymbol{\theta}_k) = \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, $\boldsymbol{\Sigma}_k = \operatorname{diag}(\boldsymbol{\sigma}_k^2)$. В этом случае ЕМ-алгоритм сводится к следующим вычислениям:

Е шаг :

$$p(z_k | \mathbf{x}_i) = \frac{p(z_k)p(\mathbf{x}_i | \boldsymbol{\theta}_k)}{\sum_{k=1}^K p(z_k)p(\mathbf{x}_i | \boldsymbol{\theta}_k)} = \frac{\pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} = \gamma_{k,i}. \quad (4.8)$$

М шаг :

$$\boldsymbol{\pi}_k^* = \frac{\sum_{i=1}^N \gamma_{k,i}}{N}, \quad \boldsymbol{\mu}_k^* = \frac{\sum_{i=1}^N \gamma_{k,i} \mathbf{x}_i}{\sum_{i=1}^N \gamma_{k,i}}, \quad \boldsymbol{\Sigma}_k^* = \frac{\sum_{i=1}^N \gamma_{k,i} (\mathbf{x}_i - \boldsymbol{\mu}_k^*)(\mathbf{x}_i - \boldsymbol{\mu}_k^*)^T}{\sum_{i=1}^N \gamma_{k,i}}. \quad (4.9)$$

Ключевое различие между моделью гауссовой смеси и кластеризацией k -средних заключается в том, что GMM учитывает не только центры кластеров, но и расстояние между кластерами и принадлежащими им векторами с априорными вероятностями для каждого кластера.

Модуль смеси слотов

Для объектно-центричного обучения предлагается модифицированный подход с использованием модели гауссовой смеси, называемый модулем смешивания слотов (SMM). Этот модуль использует из GMM шаги **E** и **M** для сопоставления карт признаков, получаемых из сверточного нейросетевого кодировщика (CNN), с набором векторных представлений слотов, где слоты представляют собой объединение средних значений распределений и диагонали ковариационной матрицы.

Как и слотовое внимание, SMM действует GRU, который использует текущие и предыдущие средние значения в качестве входных данных и скрытых состояний. В слотовом внимании представления слотов являются центрами кластеров, поэтому этот подход ограничен информацией, содержащейся и представленной в этих центрах кластеров. Рассмотрение не только средневзвешенного значения группы векторов в качестве ее представления дает преимущество в различении схожих, но все-таки разных групп. Например, два набора векторов, отобранных из разных распределений, но с одинаковым математическим ожиданием, были бы неразличимы, если бы их среднее значение было единственным представлением этих наборов.

Этот набор слотов в дальнейшем используется для решения конкретной (downstream) задачи. Также применяется тот же дополнительный шаг обновления нейронной сети для средних значений перед обновлением значений матрицы ковариаций:

$$\boldsymbol{\mu}_k = \text{RNN}(\text{input}=\boldsymbol{\mu}_k^*, \text{hidden}=\boldsymbol{\mu}_k), \quad \boldsymbol{\mu}_k = \text{MLP}(\text{LayerNorm}(\boldsymbol{\mu}_k)) + \boldsymbol{\mu}_k. \quad (4.10)$$

Эти два шага необходимы для решения последующей задачи, связывая внешнюю и внутреннюю модели. Внутренняя модель (шаги **E** и **M** в SMM) пытается обновить свои параметры $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ таким образом, чтобы входные векторы x были назначены слотам с максимальным правдоподобием. Напротив, внешняя модель принимает эти параметры в качестве входных данных. В листинге 1 представлен полный псевдокод метода SMM. Функция $f_{\theta}(x, \boldsymbol{\mu}, \boldsymbol{\Sigma}_{diag}) : \mathbb{R}^{N \times D} \times \mathbb{R}^{K \times D} \rightarrow \mathbb{R}^{N \times K}$ обозначает логарифмическую функцию плотности гауссова распределения, которая вычисляет матрицу логарифмических вероятностей, где каждый элемент представляет собой оценку логарифмической вероятности того, что вектор из x будет сгенерирован соответствующим гауссовым распределением.

Необходимо отметить, что SMM не выполняет обновления **EM** явно, поскольку алгоритм использует обновления нейронной сети, которые не обучаются максимизировать ожидания, как в традиционном EM-алгоритме. Вместо этого SMM сочетает ключевые шаги обновления из классического EM-алгоритма с обучаемыми обновлениями, специфичными для нейросетевых аппроксиматоров. Внедряя обучаемые обновления в алгоритмический цикл, предлагаемый подход использует принципы и идеи EM-алгоритма, одновременно используя преимущества гибкости и адаптивности нейронных сетей. Сочетание классических этапов EM с обучаемыми обновлениями позволяет использовать более динамичный подход,

основанный на данных, что потенциально приводит к повышению производительности и адаптивности к различным сценариям.

Algorithm 1: Псевдокод модуля смеси слотов. $\boldsymbol{\pi}$ инициализируется равномерным категориальным распределением, $\boldsymbol{\mu}$ и $\boldsymbol{\Sigma}_{diag}$ инициализируются гауссовыми распределениями с обучаемыми параметрами.

Data: $\mathbf{x} \in \mathbb{R}^{N \times D}$

```

1 — карта признаков CNN с дополнительными позиционными представлениями;
 $\boldsymbol{\mu}, \boldsymbol{\Sigma}_{diag} \in \mathbb{R}^{K \times D}, \boldsymbol{\pi} \in \mathbb{R}^K$  — параметры инициализации SMM. Result: slots
 $\in \mathbb{R}^{N \times 2D}$ 

2  $\mathbf{x} = \text{MLP}(\text{LayerNorm}(\mathbf{x}))$  for  $t = 0 \dots T$  do
3   logits =  $f_{\theta}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}_{diag})$ ;
4    $\gamma = \text{SoftMax}(\text{logits} + \log \boldsymbol{\pi}, \text{dim}=K)$ ;
5    $\boldsymbol{\pi} = \gamma.\text{mean}(\text{dim}=N)$ ;
6    $\boldsymbol{\mu}^* = \text{WeightedMean}(\text{weights}=\gamma, \text{values}=\mathbf{x})$ ;
7    $\boldsymbol{\mu} = \text{GRU}(\text{input}=\boldsymbol{\mu}^*, \text{hidden}=\boldsymbol{\mu})$ ;
8    $\boldsymbol{\mu} += \text{MLP}(\text{LayerNorm}(\boldsymbol{\mu}))$ ;
9    $\boldsymbol{\Sigma}_{diag} = \text{WeightedMean}(\text{weights}=\gamma, \text{values}=(\mathbf{x} - \boldsymbol{\mu})^2))$ ;
10 slots = concat([\boldsymbol{\mu}, \boldsymbol{\Sigma}_{diag}], dim=D);
11 return slots

```

Модуль смеси слотов можно рассматривать как расширение модуля слотового внимания (см. рисунок 4.5) со следующими ключевыми отличиями:

1. SMM обновляет не только средние значения, но также значения матрицы ковариаций и априорные вероятности,
2. вместо внимания со скалярным произведением используется функция гауссовой плотности вероятности, и
3. слоты рассматриваются не только как средние значения кластера, но и как объединение средних и ковариационных значений.

4.2.3 Экспериментальное исследование модуля смеси слотов

Поскольку SMM можно рассматривать как расширение подхода слотового внимания, предлагается использовать последний в качестве основного подхода для сравнения. SMM-подход предсказывает среднее значение $\boldsymbol{\mu}$ и матрицу ковариаций $\boldsymbol{\Sigma}_{diag}$, что удваивает размер представления промежуточного слота, поэтому применяется дополнительная матрица, чтобы эффективно уменьшить размерность представлений промежуточного слота в два раза (размер матрицы равен $2D \times D$). Таким образом, конечный размер слотов, формируемых SMM, такой же, как и у слотов, получаемых в SA. Также необходимо

отметить, что и SMM, и SA имеют одинаковое количество обучаемых параметров, поскольку SMM не использует проекции k и v (размер матриц k и v равен $D \times D$), что обеспечивает корректное сравнение этих двух подходов.

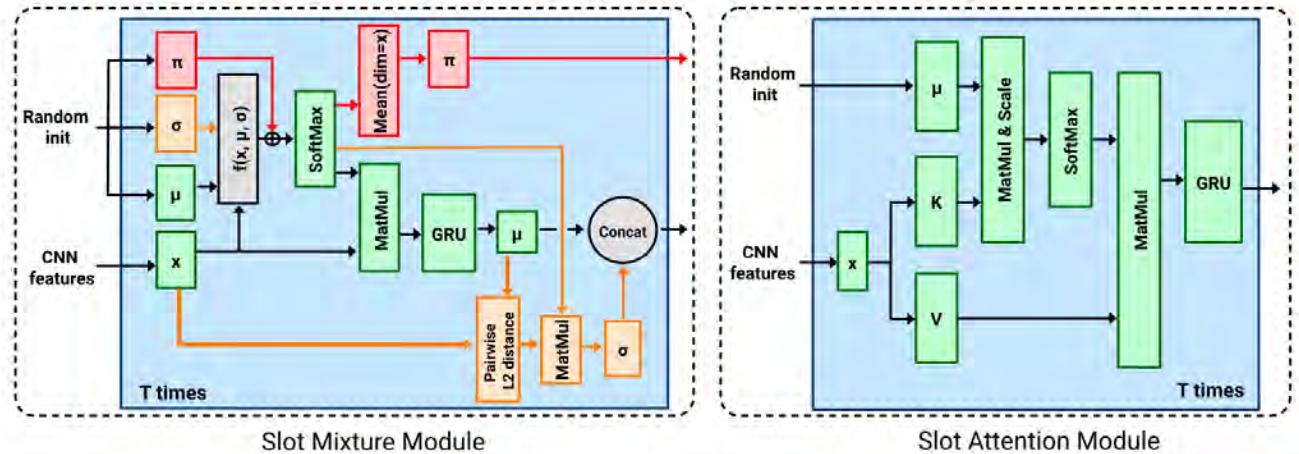


Рисунок 4.5 — Сравнение архитектур предлагаемого модуля смеси слотов (SMM) и модуля слотового внимания. Зеленый цвет используется для обозначения шагов, задействованных в обоих модулях. SMM включает оценку центров кластеров (μ), расстояния между центрами кластеров и отнесенными к ним векторами (σ , оранжевые шаги) и веса априорной смеси (π , красные шаги). Объединение μ и σ служит итоговым представлением слотов, а π может использоваться для идентификации пустых слотов, которые не содержат информации о каких-либо объектах, $f(x, \mu, \sigma)$ является логарифмом гауссовой функции плотности.

Модуль слотового внимания (SA) оценивает только центры кластеров.

В ходе экспериментов были обучены две версии одной и той же модели: одна модель с модулем слотового внимания, а другая — с предлагаемым подходом SMM, — сохраняя одинаковые условия процесса обучения. Была использована так называемая техника отсоединения слотов от градиентного вычислительного графа на последней итерации алгоритма предложенная в [174], чтобы добиться стабильного обучения и избежать взрыва градиентов. В данной работе также показано, что обучение с небольшим числом итераций (одна итерация) может привести к значительно худшей производительности в конкретных задачах, чем обучение с большим числом итераций (семь итераций). Были использованы пять итераций в качестве компромиссного варианта по умолчанию, балансирующего высокую производительность и количество шагов вычислений. Поскольку оптимальное количество шагов может зависеть от наборов данных и задач, поиск по этому гиперпараметру не выполнялся, так как это привело бы к трудностям при сравнении работы на различных задачах и наборах данных. Было выбрано среднее число в пять итераций на основе релевантных предлагаемому подходу работ: [471] использует три итерации, [573] использует семь итераций для некоторых наборов данных и три для других. [174] показывает разницу в производительности между одной и семью итерациями.

Реконструкция изображения с использованием трансформерной модели

Была использована базовая модель SLATE [573], в которой модуль SA заменен на SMM, для решения задачи сравнения изображений без учителя. SLATE использует декодировщик изображений на базе Image GPT [277], обусловленный на представления слотов, для авторегрессионного восстановления дискретных визуальных токенов из дискретного вариационного автокодировщика (dVAE) [222]. Он обрабатывает предварительно вычисленные представления слотов как векторы запросов, а векторы скрытого кода изображения — как векторы ключа/значения. Несмотря на то, что SLATE очень качественно фиксирует сложные соотношения между слотами и пикселями в синтетических изображениях, его производительность снижается при работе с более реалистичными данными. Функцией потерь процесса обучения кодировщика dVAE, декодировщика и скрытых дискретных токенов является среднеквадратичная ошибка MSE между входным и восстановленным изображениями. Модули SA/SMM и Image GPT обучаются с функцией потерь в виде перекрестной энтропии с использованием сжимания изображений в токены dVAE в качестве целевого распределения. Этот процесс изолирован от остальной части модели (т.е. от dVAE), но обе подсистемы обучаются одновременно.

Были рассмотрены следующие наборы данных: CLEVR-Mirror [573], ClevrTex [348], ShapeStacks [562] и COCO-2017 [432]. CLEVR-Mirror является расширением стандартного набора данных CLEVR, который требует определения глобальных связей между локальными компонентами из-за наличия зеркального отражения объектов. ShapeStacks проверяет способность модели описывать сложные локальные соотношения признаков (множество объектов, расположенных друг над другом), а ClevrTex исследует возможности модели в сценах с богатой текстурой. Для ShapeStacks, ClevrTex и COCO были использованы изображения, отмасштабированные до разрешения 96×96 , а изображения CLEVR-Mirror масштабируются до 64×64 . Условия обучения с гиперпараметрами, соответствующими определенному набору данных, выбраны аналогично работе [573] за исключением того, что использовался размер пакета данных равный 64 и для всех экспериментов была отсечка на $2,5 \times 10^5$ итерациях.

Таблица 10 показывает основные метрики производительности восстановления изображения для тестовых частей описанных выше наборов данных. Начальное расстояние Фреше (FID) [269] вычислялось с помощью библиотеки PyTorch-Ignite³ для оценки качества сгенерированных изображений. Были также оценены перекрестная энтропия между токенами, полученными из dVAE (в виде унитарного (one-hot) распределения), и распределением, прогнозируемым с помощью Image GPT, обусловленного определенными представлениями слотов, поскольку значение FID может быть ограничено величиной, полученной только с дискретным автокодировщиком.

³<https://github.com/pytorch/ignite>

Таблица 10 — Качество реконструкции (среднее значение \pm std для четырех запусков). SLATE(SA) — оригинальная модель SLATE, SLATE(SMM) — модифицированный SLATE с SMM вместо SA. В качестве верхней оценки метрики приведены результаты FID для dVAE.

Data	FID		Cross-Entropy		
	SLATE(SA)	SLATE(SMM)	dVAE	SLATE(SA)	SLATE(SMM)
CLEVR-Mirror	35.4 ± 1.1	34.8 ± 0.9	32.2 ± 0.9	0.82 ± 0.05	0.20 ± 0.02
ShapeStacks	56.6 ± 2.8	50.4 ± 1.84	40.4 ± 1.3	88.3 ± 2.2	62 ± 2.1
ClevrTex	116 ± 4.6	113 ± 4.3	82.8 ± 3.9	566 ± 18.1	507 ± 21.7
COCO	129 ± 5.2	122 ± 4.1	92.0 ± 4.4	540 ± 17.5	461 ± 12.1
CelebA	39.2 ± 1.5	41.3 ± 1.6	35.7 ± 1.1	734 ± 31.9	655 ± 28.6
Bitmoji	30.8 ± 0.52	27.0 ± 0.57	25.1 ± 0.4	15.1 ± 0.16	13.3 ± 0.21

SLATE состоит из двух частей, которые обучаются раздельно: dVAE и авторегрессионная трансформерная модель. Трансформерная модель принимает представления слотов в качестве входных данных и стремится реконструировать токены из кодировщика dVAE, минимизируя перекрестную энтропию между истинными (ground truth) токенами и предсказанными токенами. Следовательно, более низкое значение перекрестной энтропии рассматривалось как свидетельство более высокой выразительности представлений SMM по сравнению с представлениями SA, поскольку это помогает точно реконструировать исходные токены. Лучшее предсказание токенов, закодированных в dVAE, приводит к повышению качества восстановления изображения, даже несмотря на то что эта модель не была непосредственно обучена для этого, поскольку только dVAE был обучен путем минимизации функции потерь по восстановлению изображения.

В таблице 10 представлены результаты для модели SLATE и ее модификации с помощью SMM. SLATE не выводит маски объектов, как это делает модель обнаружения объектов SA из статьи [471], так как SLATE использует карты внимания в качестве эвристики для визуализации информации о каждом слоте. Было бы неверно рассматривать их как маски для вычисления метрики FG-ARI, поскольку данная задача не подразумевает обнаружения объектов.

На рисунках 4.6 и 4.7 представлены примеры восстановленных изображений из наборов данных ShapeStacks, ClevrTex и CLEVR-Mirror. Качественно можно заметить способность SMM точнее воссоздавать правильный порядок объектов по исходным изображениям.

Прогнозирование множества свойств

Нейросетевые модели для анализа множеств используются в различных приложениях во многих модальностях обработки данных [248; 249; 392; 570]. Прогнозирование множества свойств — это задача обучения с учителем, которая предполагает, что обученная модель

предсказывает неупорядоченный набор векторов, представляющих свойства объектов входного изображения. Множества предсказанных и целевых векторов сопоставляются с использованием венгерского алгоритма [368], а функция потерь для обучения реализуется через функцию Хубера [659] между сопоставляемыми векторами. Модуль смешивания слотов подходит для работы с множествами свойств, поскольку он сохраняет эквивариантность перестановок относительно компонентов смеси (слотов) и инициализирует их случайным образом.

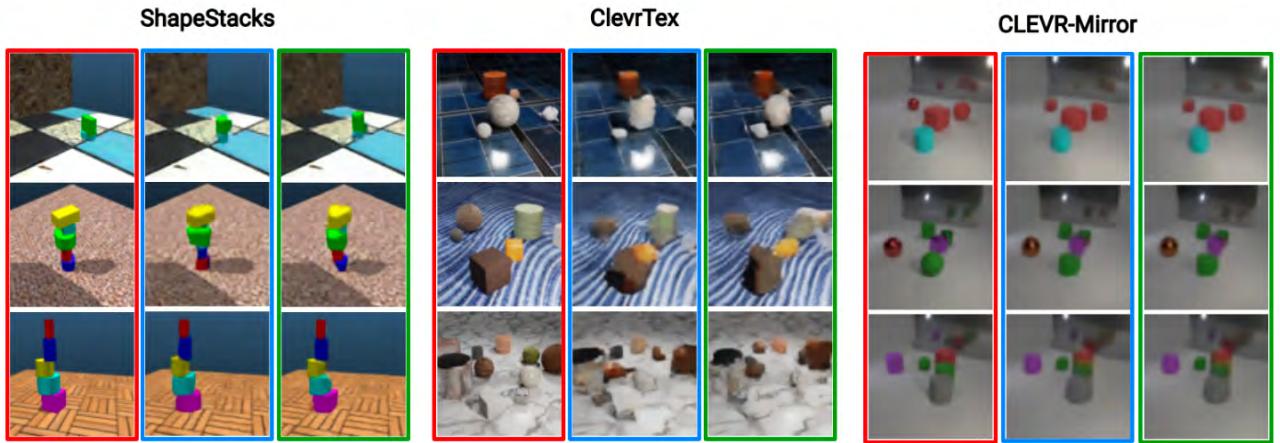


Рисунок 4.6 — Примеры генерации изображений с использованием Image GPT, обусловленные представлениями различных слотов. Изображения с синими границами взяты из модели с модулем слотового внимания, а изображения с зелеными границами сгенерированы с использованием слотов из модуля смеси слотов. Красный цвет границы обозначает исходные изображения.

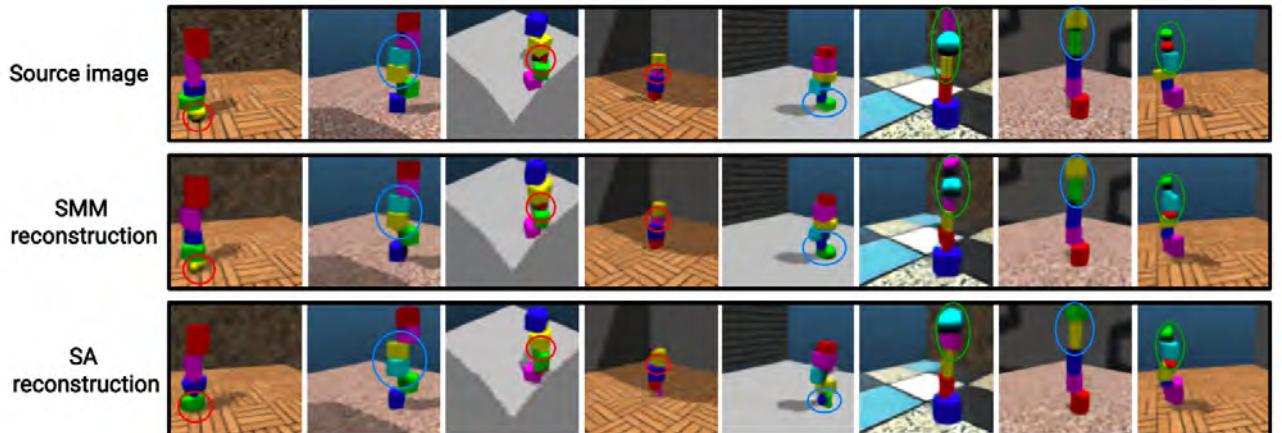


Рисунок 4.7 — Примеры всех качественно некорректно сгенерированных изображений из случайного пакета размером 64 примеров. В шести случаях реконструкция с использованием слотового внимания привела к неправильному порядку объектов (синий круг) или потере одного объекта (красный круг). В оставшихся двух примерах оба модуля имеют неправильную реконструкцию (зеленый круг).

В ходе экспериментов был использован набор данных CLEVR, поскольку он является стандартным для сравнения объектно-центрических моделей в задаче прогнозирования множества свойств. В качестве непосредственных данных для обучения выступали отмасштабированные до разрешения 128×128 изображения из этого набора данных. Все модули обучаемой модели, кроме SA/SMM, такие же, как в работе [471]. Каждая модель обучается с помощью аддитивного оптимизатора Adam [352] за $1,5 \times 10^5$ итераций с циклическим расписанием на скорость обучения OneCycleLR [576] при максимальной скорости обучения в 4×10^{-4} . Использовался размер пакета данных в размере 512. Количество итераций SA/SMM устанавливается равным 5 во время обучения и 7 во время оценки. Количество слотов равно 10, поскольку изображения CLEVR содержат не более 10 объектов.

Табл.11 демонстрирует количественные результаты экспериментов, в которых вычислялась средняя точность (AP) с различными пороговыми значениями на расстояние для прогнозирования свойств объекта, представляющих координаты местоположения. Полученные результаты показывают, что модификации исходного слотового внимания, такие как отсоединение слотов перед заключительной итерацией уточнения и изменение масштаба координат, могут повысить его производительность. Однако он по-прежнему отстает от самой производительной модели iDSPN. При нестрогих пороговых значениях другие модификации SA (SA-MESH [630]) также дают почти идеальные результаты. Напротив, SMM неизменно превосходит узкоспециализированную модель, такую как iDSPN, и достигает наиболее высоких результатов при строгих пороговых значениях (начиная с 0,25).

Таблица 11 — Качество работы в задаче прогнозирования множества свойств для набора данных CLEVR (AP в %, среднее значение \pm std для четырех запусков в экспериментах). Для слотового внимания [471] и для iDSPN [659] результаты взяты из оригинальных статей. SA* — это модель слотового внимания, обученная с теми же условиями, что и SMM: отсоединенные слоты на последней итерации с масштабированием координат до диапазона [-1, 1]. Результаты SA-MESH взяты из оригинальной статьи [630].

Model	AP _{∞}	AP ₁	AP _{0.5}	AP _{0.25}	AP _{0.125}	AP _{0.0625}
SA	94.3 ± 1.1	86.7 ± 1.4	$56.0 \pm 3.$	10.8 ± 1.7	0.9 ± 0.2	—
SA*	97.1 ± 0.7	94.5 ± 0.7	88.3 ± 3.2	62.5 ± 5.4	23.6 ± 1.4	4.6 ± 0.3
SA-MESH	99.4 ± 0.1	99.2 ± 0.2	98.9 ± 0.2	91.1 ± 1.1	47.6 ± 0.8	12.5 ± 0.4
iDSPN	98.8 ± 0.5	98.5 ± 0.6	98.2 ± 0.6	95.8 ± 0.7	76.9 ± 2.5	32.3 ± 3.9
SMM (ours)	99.4 ± 0.2	99.3 ± 0.2	98.8 ± 0.4	98.4 ± 0.7	92.1 ± 1.2	47.3 ± 2.5

Таблица 12 — Значения метрики FG-ARI для набора данных ClevrTex.

Dataset	SA	SMM
ClevrTex	62.40 ± 2.33	74.7 ± 1.3



Рисунок 4.8 — Пример обнаружения объекта в наборе данных ClevrTex. Первый столбец представляет исходные истинные изображения, второй — реконструкции широковещательного декодировщика. Последующие столбцы соответствуют слотам масок внимания.

Обнаружение объектов

Другой объектно-центричной задачей класса «изображение-изображение» с обучением без учителя является задача обнаружения объектов. Для нее использовалась модель из работы [471]. Этот подход декодирует представление каждого слота в четырехканальное изображение с использованием пространственного широковещательного декодировщика (spatial broadcast decoder) [585]. Результирующая реконструкция в пиксельном пространстве оценивается как смесь декодированных слотов, где первые три канала отвечают за восстановленное изображение RGB, а четвертый канал — за веса компонентов смеси, которые интерпретируются как маски объектов.

Поскольку SA показывает почти идеальные показатели на таких простых наборах данных как CLEVR были проведены эксперименты на сложном наборе данных — ClevrTex. Была рассмотрена та же настройка процесса обучения, что и в оригинальной работе [471] для CLEVR. Единственное отличие заключается в том, что разрешение изображения ClevrTex уменьшено до 64×64 .

Табл. 12 показывает сходство между истинными масками сегментации объектов (за исключением фона) и весовыми коэффициентами смеси, оцененными с помощью показателя Rand с поправкой на передний план (FG-ARI) в ClevrTex (указано среднее значение \pm std для трех запусков эксперимента). SMM существенно превосходит SA в таком сложном наборе данных. Примеры обнаружения объектов с помощью SMM показаны на рисунке 4.8.

Были также проведены эксперименты с более простыми наборами данных, такими как CLEVR10, Multi-dSprites и Tetromino. Поскольку SA показывает почти идеальные результаты при достаточно длительном обучении на этих наборах данных, были сокращены количество шагов обучения до 300 тысяч (вместо 500 тысяч), чтобы сравнить скорость сходимости двух подходов. Результаты FG-ARI следующие: i) CLEVR10 — SA $85,6 \pm 1,2$, SMM $91,3 \pm 0,6$; ii) Мульти-dSprites — SA $81,0 \pm 0,7$, SMM $89,6 \pm 0,6$; iii) Тетроминоны — SA $75,4 \pm 1,0$, SMM $83,2 \pm 0,7$. Метод SMM показывает более быструю сходимость, что проявляется в более качественной сегментации объектов.

Дополнительные эксперименты

Основываясь на подходе библиотеки концептов, представленном в [573], было предложено его расширение, в котором коллекция слотов улучшается за счет отбрасывания пустых слотов. Это усовершенствование возможно благодаря тому, что в SMM подсчитывается оценка априорных весов смеси π . К этому оптимизированному набору была применена кластеризация GMM, которая сохраняет возможность назначать слоты кластерам, идентифицировать базовые концепты и осуществлять выборку концептов благодаря оценке параметров гауссовского распределения.

Предлагаемый подход к оценке качества редактирования изображений включает в себя извлечение и выборку новых слотов для каждого изображения из валидационного набора данных, используя гауссовые параметры соответствующих им кластеров. Эти отобранные слоты затем вводятся в декодировщик для генерации результирующего изображения. Эксперименты с наборами данных Bitmoji, ShapeStacks и CLEVR-Mirror дали оценки значения FID в размере 32,1, 59 и 41,4, соответственно, что указывает на высокое качество отредактированных изображений, сравнимое с обычным автокодировщиком (таблица 11). Рисунок 4.9 иллюстрирует такую выборку концептов, примененную к одному слоту.

Модули слотового внимания и SMM могут быть сведены соответственно к подходам кластеризации k -средних и гауссовой модели смеси путем удаления обновлений GRU/MLP, обучаемых проекций q, k, v и слоев послойной нормализации [142]. Табл. 13 показывает результаты в задаче прогнозирования множества свойств для набора данных CLEVR с использованием таких упрощенных методов кластеризации. Проведенные эксперименты демонстрируют, что кластеризация GMM лучше подходит для объектно-центричного обучения даже без обучаемых слоев.

В данном разделе диссертационного исследования представлен модуль смешивания слотов, который является новым слотовым подходом к задаче формирования объектно-центричных представлений, обобщающий модель гауссовой смеси. Предложенный подход учитывает центры кластеров и расстояния между кластерами для улучшения представления слотов.

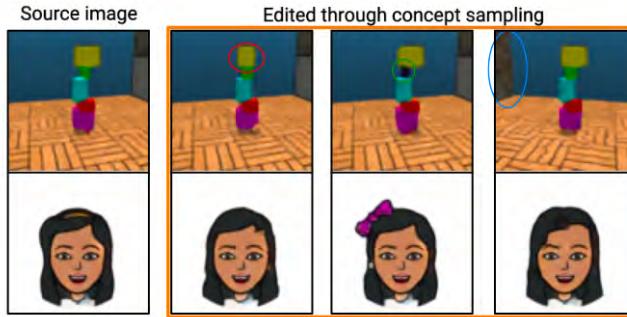


Рисунок 4.9 — Пример редактирования изображений из набора данных Shape Stacks и Bitmoji с использованием выборки концептов. Этот метод позволяет точно редактировать отдельные концепты: в первой строке показана возможность изменения формы, цвета отдельных объектов или манипуляция с фоном. Во второй строке приведен пример создания похожих, но немного отличающихся причесок с помощью выборки концептов.

Таблица 13 — Средняя точность (среднее значение $\pm \text{std}$ для четырех запусков эксперимента) с различными пороговыми значениями расстояния для задачи прогнозирования множества свойств в наборе данных CLEVR после 100 тысяч итераций обучения.

Model	AP_{∞}	AP_1	$\text{AP}_{0.5}$	$\text{AP}_{0.25}$	$\text{AP}_{0.125}$
K-means	81.7 ± 1.6	49.1 ± 2.9	7.2 ± 1.7	1.4 ± 0.3	0.2 ± 0.1
GMM	88.6 ± 2.0	53.3 ± 2.2	9.2 ± 1.6	2.3 ± 0.3	0.5 ± 0.1

Эмпирические данные для набора данных CLEVR [179] подтверждают наилучшую производительность SMM-модели в задаче прогнозирования множества свойств со строгими пороговыми значениями и высокую производительность в задаче реконструкции изображений. Кроме того, были продемонстрированы улучшенная производительность реконструкции на синтетических наборах данных, таких как CLEVR-Mirror [573], ShapeStacks [562] и ClevrTex [348]. SMM также превосходит SA при использовании сложного набора данных, такого как ClevrTex, в задаче обнаружения объектов. Однако ограниченная эффективность формирования объектных представлений современными моделями [161] на реальных изображениях, как, например, COCO-17 [432], подчеркивает необходимость продолжения исследований в этом области.

4.3 Обучение с подкреплением с объектно-центричными моделями

В данном разделе диссертационного исследования рассматривается возможность использования объектного представления для обучения актора и критика в архитектуре NSLP без использования иерархии умений (рисунок 4.10). Исследованы особенности использования слотовой нейросимвольной объектной модели $NS(o_t)$ с графовым

представлением G_t для обучения стратегии агента. Основные результаты, изложенные в данном разделе, представлены в публикациях [55; 58].

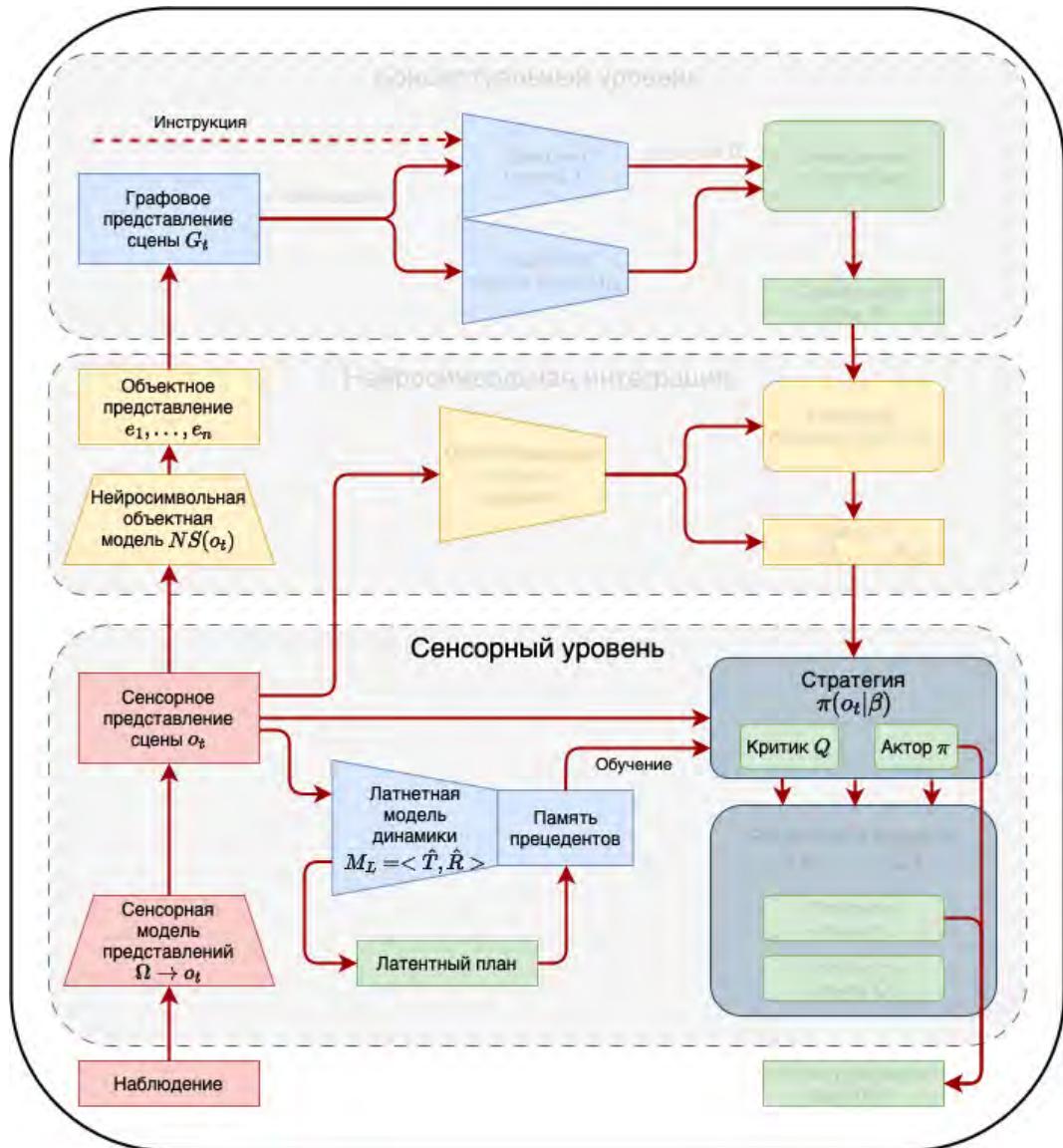


Рисунок 4.10 — Архитектура NSLP с выделенным нейросимвольным уровнем, на котором формируется объектное представление для обучения стратегии агента с использованием модели среды.

4.3.1 Объектно-центрические подходы для построения модели мира

Одной из основных проблем в обучении с подкреплением в визуальных средах является определение того, как эффективно представлять состояние окружающей среды. Наиболее распространенным подходом является кодирование всего входного изображения, которое затем используется в качестве входных данных для нейросетевого аппроксиматора стратегии [323; 385]. Однако предыдущие исследования [109] показали, что такие

представления могут не отражать значимые отношения и взаимодействия между объектами в состоянии. Для решения этой проблемы, как уже указывалось в предыдущих разделах диссертационного исследования, могут быть введены объектно-центричные представления. Предполагается, что в результате использования таких представлений будут получены более компактные модели с расширенными возможностями генерализации [258]. Современные модели объектно-центричного представления (OCR), обучаемые без учителя [252; 471; 573], имеют практическую значимость с точки зрения обучения с подкреплением, поскольку они не требуют дополнительной разметки данных для обучения. Недавние исследования [129; 394] показали, что объектно-центричная факторизация состояний может улучшить способность безмодельных алгоритмов к обобщению и повысить эффективность выборки из среды (sample efficiency).

Другим способом сократить количество необходимых для обучения примеров из среды является использование методов, основанных на модели среды [532]. Как было показано в разделе 1.3, в обучении с подкреплением на основе модели среды (MBRL) агент строит динамические аппроксимации для функций перехода и вознаграждения на основе своего опыта взаимодействия с окружающей средой. Агент выполняет многошаговое планирование, чтобы выбрать оптимальное действие, используя только предсказания модели, без взаимодействия с окружающей средой. Алгоритмы, основанные на модели среды, могли бы быть более эффективными, чем безмодельные алгоритмы, если бы точность модели мира была достаточной. Современные методы MBRL, использующие обучение в «воображении» [421] и предварительный поиск с использованием модели динамики, эквивалентной по полезности [418], позволяют проводить эффективное обучение в широком наборе сред.

Для дальнейшего повышения эффективности выборки из среды многообещающим направлением является объединение обоих подходов путем разработки модели мира, которая использует объектные представления и явно обучается моделировать отношения между объектами [55]. Примером такого подхода является модель переходов с контрастивным обучением CSWM [354]. Эта модель использует графовую нейронную сеть для аппроксимации динамики окружающей среды и одновременно обучается факторизовать состояние и прогнозировать изменения в состоянии отдельных объектов. Алгоритм CSWM продемонстрировал наилучшее качество прогнозирования по сравнению с традиционными монолитными моделями.

Необходимо отметить, что объектно-центричные модели демонстрируют высокое качество в относительно простых средах с хорошо различимыми объектами [575]. Кроме того, в объектных средах действия часто применяются к одному объекту или к небольшому количеству объектов, что упрощает прогнозирование динамики отдельных объектов. В более сложных средах модель мира должна точно связывать действия с объектами, к которым они применяются, чтобы эффективно предсказывать переходы в среде. Несмотря на недавний прогресс в этой области [155], к настоящему моменту не было предложено ни одной полнофункциональной модели динамики, которая бы учитывала разреженность отношений типа действие-объект. Это обстоятельство затрудняет использование объектно-центрических

моделей мира в обучении с подкреплением. Например, модель CSWM не использовалась для обучения стратегии агента ни в автономном режиме, ни в режиме «онлайн».

Предлагаемый в данном разделе метод обучения с подкреплением предполагает использование латентной модели мира и использует подход с явным восстановлением функции полезности, поскольку ее объектно-центрическая декомпозиция может способствовать обучению самой объектно-центрической модели мира, согласующейся со стратегией агента. В соответствии с этим подходом разработан графовый объектно-центрический актор-критик (ROCA), использующий отложенный опыт для обучения стратегии на базе объектно-центрической модели представлений наблюдений и основывающийся на энтропийной регуляризации мягкого актора-критика (SAC) [176; 580; 581]. Предлагаемый метод работает как с дискретными, так и с непрерывными пространствами действий.

Алгоритм ROCA использует предварительно обученную модель SLATE [573], которая извлекает представления отдельных объектов из входного изображения. Аналогично CSWM [354] используется структурированная модель переходов, основанная на графовых нейронных сетях (GNN). Обученные модели вознаграждения, полезности состояния и актора представляют собой графовые нейронные сети, согласованные с объектно-центрической структурой задачи. В соответствии с архитектурой TreeQN [624] в предлагаемом алгоритме используется модель мира в компоненте критика для прогнозирования полезности действий.

Алгоритм ROCA является первым алгоритмом, в котором успешно применена объектно-центрическая модель мира на основе GNN для обучения стратегии в постановке обучения с подкреплением. С целью оценки качества предложенного алгоритма были проведены эксперименты в 2D-средах с объектами простой формы и визуально более сложными симуляционными робототехническими 3D-средами. Предлагаемый алгоритм демонстрирует высокую эффективность выборки из среды и превосходит объектно-центрический вариант безмодельного алгоритма PPO [512], который использует ту же модель SLATE в качестве способа извлечения признаков и построен на трансформерной архитектуре. Кроме того, предложенный метод работает лучше, чем современный MBRL-алгоритм DreamerV3 [421].

Недавние достижения в области компьютерного зрения были, в том числе, связаны с созданием алгоритмов обучения без учителя для объектно-центрического представления сцен [252; 471; 657]. Эти методы направлены на изучение структурированных визуальных представлений на основе изображений без использования размеченных наборов данных за счет моделирования каждого изображения в виде композиции объектов. Это направление исследований диктуется его потенциальными преимуществами для различных конечных задач анализа изображений, в том числе потенциально более высоким уровнем генерализации и способностью моделировать рассуждения о свойствах визуальных объектов. Одним из основных подходов в этой области является алгоритм слотового внимания [471] (см., в том числе, предыдущие разделы), который представляет объекты с использованием нескольких скрытых переменных и уточняет эти представления с помощью механизма внимания. Основываясь на этом подходе, метод SLATE [657] позволяет повысить

эффективность выделения объектов за счет декодировщика на основе трансформерной модели вместо декодировщика, использующего смесь на уровне пикселей изображения.

В работе [394] слотовое внимание используется в качестве объектно-центричного подхода к извлечению признаков и исследуется производительность методов обучения с подкреплением и возможности их генерализации. В другом исследовании [470] предлагается многоступенчатый подход к обучению, включающий дообучение модели YOLO⁴ на наборе данных, размеченных с помощью объектно-ориентированной модели, обученной без учителя. Зафиксированная модель YOLO затем используется в качестве объектно-центричного подхода к извлечению признаков в алгоритме дуэльного DQN. Представления объектов объединяются с использованием нейросетевой графовой модели внимания перед передачей в Q-сеть.

В качестве связанной с настоящим исследованием работы в объектно-центричном MBRL необходимо указать [184]. В ней используется модель MONet [444] в качестве объектно-центричного подхода к извлечению объектов и исследуется объектно-центричная модель перехода в среде. Однако, в отличие от предлагаемого подхода ROCA, эта модель не учитывает взаимодействия между объектами и используется только на этапе исследования среды в базовом алгоритме обучения с подкреплением.

4.3.2 Графовый объектный актор-критик

В данном разделе в качестве базового алгоритма обучения с подкреплением используется так называемый мягкий актор-критик (SAC) [580; 581], современный алгоритм обучения подкреплением на основе отложенного опыта для формирования стратегии с использованием непрерывных действий.

Целевая функцию при обучении этого алгоритма направлена на поиск стратегии, которая максимизирует энтропийный целевой функционал:

$$\pi^* = \arg \max_{\pi} \sum_{i=0}^{\tau} \mathbb{E}_{(s_t, a_t) \sim d_{\pi}} [\gamma^t (R(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot | s_t)))] \quad (4.11)$$

где α — параметр температуры, $\mathcal{H}(\pi(\cdot | s_t)) = -\log \pi(\cdot | s_t)$ — энтропия стратегии π в состоянии s_t , d_{π} — распределение траекторий, индуцированное стратегией π .

Функция мягкой полезности действий $Q_{\theta}(s_t, a_t)$, параметризованная с использованием нейронной сети с параметрами θ , обучается путем минимизации сглаженного оператора Беллмана:

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t) \sim D} [(Q_{\theta}(s_t, a_t) - R(s_t, a_t) - \gamma \mathbb{E}_{s_{t+1} \sim T(s_t, a_t)} V_{\bar{\theta}}(s_{t+1}))^2], \quad (4.12)$$

где D — память прецедентов взаимодействия агента со средой (прошлый опыт агента), а $V_{\bar{\theta}}(s_{t+1})$ оценивается с использованием целевой сети для функции Q и Монте-Карло оценки сглаженной функции полезности состояния после выборки прецедентов из памяти D .

⁴<https://github.com/ultralytics/yolov5>

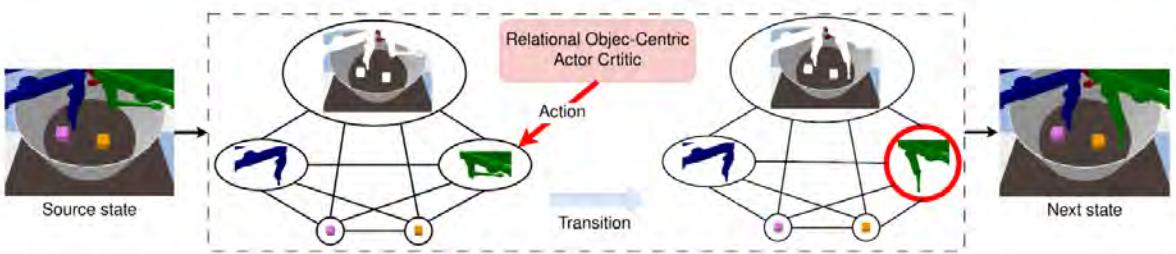


Рисунок 4.11 — Схема работы предлагаемого подхода ROCA, в котором модель формирует стратегию, извлекая объектно-центричные представления из исходного изображения и обрабатывая их в виде полного графа.

Стратегия π параметризуется с использованием нейронной сети с параметрами φ . Параметры обучаются путем минимизации ожидаемого расстояния Кульбака-Лейблера (KL-дивергенции) между стратегией и экспонентой от Q -функции:

$$J_\pi(\varphi) = \mathbb{E}_{s_t \sim D} [\mathbb{E}_{a_t \sim \pi_\varphi(\cdot|s_t)} [\alpha \log(\pi_\varphi(a_t|s_t)) - Q_\theta(s_t, a_t)]] . \quad (4.13)$$

Целевая функция для параметра температуры задается с помощью следующего выражения:

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t)} [-\alpha(\log \pi(a_t|s_t) + \bar{H})] , \quad (4.14)$$

где \bar{H} — гиперпараметр, интерпретируемый как целевая энтропия. На практике поддерживаются две отдельных обучаемых сглаженных Q -сети, а затем минимальное значение, полученное от этих двух аппроксиматоров, используется в качестве выходного результата подсчета сглаженной Q -сети.

В то время как оригинальная версия SAC решает задачи с непрерывным пространством действий, версия для дискретных пространств действий была предложена в работе [176]. В случае дискретного пространства действий стратегия $\pi_\varphi(a_t|s_t)$ выдает вероятность для действий вместо плотности вероятности. Такая параметризация стратегии незначительно изменяет целевые функционалы 4.12, 4.13 и 4.14.

На рисунке 4.11 представлена верхнеуровневая схема работы предлагаемой архитектуры актора-критика (ROCA). В качестве кодировщика предлагается использовать SLATE [573], недавно представленную объектно-центричную модель. SLATE включает в себя дискретный вариационный кодировщик [479] для извлечения представлений объектов, GPT-подобную трансформерную модель [657] для декодирования и модуль слотового внимания [471] для группировки признаков, связанных с одним и тем же объектом. В ROCA предварительно обученная модель SLATE с фиксированными параметрами принимает в качестве входных данных наблюдение в виде изображения s_t и создает набор объектных векторов, называемых слотами, $\mathbf{z}_t = (z_t^1, \dots, z_t^K)$ (K — максимальное количество извлекаемых объектов).

Модель актора реализует текущую версию стратегии агента и возвращает действие для сформированного текущего состояния \mathbf{z}_t . Критик предсказывает полезность $Q(\mathbf{z}_t, a)$ выполняемого действия a , полученного от актора с учетом текущего представления состояния \mathbf{z}_t . Данное представление оценивается с использованием обученной модели

переходов в среде, модели вознаграждения и модели полезности состояния. Представление входного состояния $\mathbf{z}_t = (z_t^1, \dots, z_t^K)$ обрабатывается компонентами ROCA, реализованными на основе GNN.

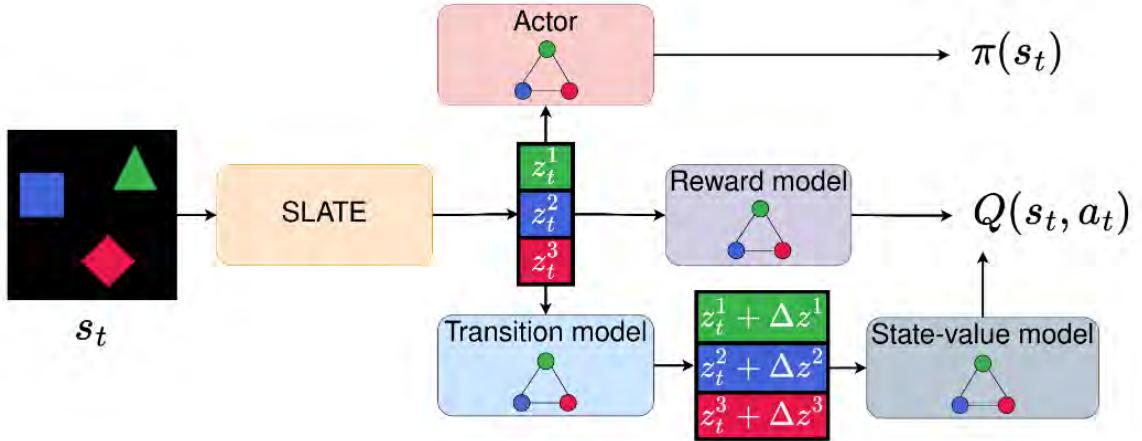


Рисунок 4.12 — Схема взаимосвязи компонент ROCA. Метод состоит из предварительно обученной модели SLATE с зафиксированными весами, которая извлекает объектно-центрические представления из наблюдения в виде изображений, и модулей на основе GNN: модели перехода, модели вознаграждения, модели полезности состояния и модели актора. Модели перехода и вознаграждения формируют модель мира. Модель мира и модель полезности состояния вместе составляют модуль критика, который предсказывает Q-значения.

В ROCA используется аппроксимация функций переходов на основе графовой нейронной сети [354] с реберной моделью edge_T и узловой моделью node_T , которая принимает в качестве входных данных факторизованное состояние $\mathbf{z}_t = (z_t^1, \dots, z_t^K)$ и действие a_t и предсказывает изменение факторизованного состояния Δz . Действие передается в узловую node_T и реберную edge_T модели, как показано на рисунке 4.12. Факторизованное представление следующего состояния получается с помощью операции сложения $\hat{\mathbf{z}}_{t+1} = \mathbf{z}_t + \Delta z$. Поскольку набор слотов рассматривается как полный граф, сложность правила обновления 4.15 квадратична по количеству слотов. То же самое относится ко всем моделям на базе GNN в ROCA:

$$\Delta z^i = \text{node}_T(z_t^i, a_t^i, \sum_{i \neq j} \text{edge}_T(z_t^i, z_t^j, a_t^i)). \quad (4.15)$$

Модель вознаграждения использует практически ту же архитектуру, что и модель переходов в среде. Отличие заключается в том, что применяется усреднение векторных представлений объектов, возвращаемых узловой моделью, а результат передается в MLP для получения скалярного вознаграждения. Модель вознаграждения обучается с использованием среднеквадратичной функции потери с вознаграждениями r_t из среды в качестве целевых значений:

$$\begin{cases} \text{embed}_R^i = \text{node}_R(z_t^i, a_t^i, \sum_{i \neq j} \text{edge}_R(z_t^i, z_t^j, a_t^i)), \\ \hat{R}(z_t, a_t) = \text{MLP}(\sum_{i=1}^K \text{embed}_R^i / K). \end{cases} \quad (4.16)$$

Функция полезности состояния аппроксимируется с использованием графовой нейронной сети \hat{V} , которая не зависит от действий ни в реберной модели edge_V , ни в узловой модели node_V . Как и в модели вознаграждения, векторные представления объектов, возвращаемые узловой моделью, усредняются и передаются в MLP для получения скалярного значения:

$$\begin{cases} \text{embed}_V^i = \text{node}_V(z_t^i, \sum_{i \neq j} \text{edge}_V(z_t^i, z_t^j)), \\ \hat{V}(\mathbf{z}_t) = \text{MLP}(\sum_{i=1}^K \text{embed}_V^i / K). \end{cases} \quad (4.17)$$

Модель актора задействует ту же архитектуру GNN, что и модель полезности состояния, но использует разные реализации выходов MLP для непрерывного и дискретного пространств действий. В случае непрерывного пространства действий персептрон возвращает среднее значение и ковариацию гауссовского распределения. Для дискретного пространства действий он выводит вероятности для всех действий:

$$\begin{cases} \text{embed}_{\text{actor}}^i = \text{node}_{\text{actor}}(z_t^i, \sum_{i \neq j} \text{edge}_{\text{actor}}(z_t^i, z_t^j)), \\ \mu(\mathbf{z}_t) = \text{MLP}_\mu(\sum_{i=1}^K \text{embed}_{\text{actor}}^i / K), \\ \sigma^2(\mathbf{z}_t) = \text{MLP}_{\sigma^2}(\sum_{i=1}^K \text{embed}_{\text{actor}}^i / K), \\ \pi(\mathbf{z}_t) = \text{MLP}_\pi(\sum_{i=1}^K \text{embed}_{\text{actor}}^i / K). \end{cases} \quad (4.18)$$

В критике применяется модель мира для прогнозирования полезности действий. В частности, используется декомпозиция Q-функции на основе уравнения Беллмана. Первоначально такая декомпозиция была введена в алгоритме обучения на основе функции полезности TreeQN [624]:

$$\hat{Q}(\mathbf{z}_t, a_t) = \hat{R}(\mathbf{z}_t, a_t) + \gamma \hat{V}(\mathbf{z}_t + \Delta \mathbf{z}), \quad (4.19)$$

где \hat{R} — модель вознаграждения 4.16, \hat{V} — модель полезности состояния 4.17, $\mathbf{z}_t + \Delta \mathbf{z}$ — предсказание следующего состояния, сгенерированное моделью переходов 4.15. Поскольку выходные полезности критика вычисляются с использованием модели мира, ROCA является методом, основанным одновременно на модели среды и на полезности состояния.

Модель SLATE предварительно обучена на наборе данных траекторий, собранных с использованием случайной стратегии (100 тыс. наблюдений для задач Shapes2D и 200 тыс. наблюдений для задачи достижения объекта). В соответствии с оригинальной работой [573] применяется уменьшение значения температуры обучения dVAE τ с 1,0 до 0,1 и «прогрев» скорости обучения для параметров кодировщика слотового внимания и трансформаторной модели в начале обучения. После предварительного обучения параметры модели SLATE сохраняются зафиксированными.

Для обучения всех остальных компонентов ROCA использует функции потерь базового алгоритма SAC: 4.12, 4.13, 4.14. Как для непрерывных, так и для дискретных сред в модуле критика применяется обычная архитектура двойной Q-сети. Кроме того, используются данные, выбранные из памяти прецедентов, для обучения компонентов модели мира. Модель переходов обучается с использованием среднеквадратичной

функции потерь, чтобы минимизировать ошибку прогнозирования представлений объектов для следующего состояния с учетом действия агента. Модель вознаграждения также обучается с использованием среднеквадратичной функции потерь относительно истинных вознаграждений из среды r_t в качестве целевых значений:

$$J_{WM} = \mathbb{E}_{s_t, a_t, r_t, s_{t+1} \sim D} [\beta_T \|\mathbf{z}_t + \Delta\mathbf{z} - \mathbf{z}_{t+1}\|^2 + \beta_R (\hat{R}(\mathbf{z}_t, a_t) - r_t)^2]. \quad (4.20)$$

В общей сложности алгоритм ROCA насчитывает четыре оптимизатора. Параметр температуры, актор и модель полезности используют отдельные оптимизаторы. Модели переходов и вознаграждения задействуют оптимизатор модели мира.

Из-за стохастической природы модели SLATE объектно-центричные представления могут перемешиваться на каждом шаге. Чтобы обеспечить соблюдение порядка представления объектов во время оптимизации целевой функции модели мира 4.20, слоты модели SLATE для следующего состояния \mathbf{z}_{t+1} предварительно инициализируются текущими значениями \mathbf{z}_t .

4.3.3 Экспериментальное исследование графового объектного актора-критика

Эффективность работы предложенного алгоритма ROCA была продемонстрирована в трехмерной робототехнической симуляционной среде CausalWorld [171] для задачи достижения объекта, как это было аналогично сделано в работе [129], и в синтетической двумерной среде Shapes2D [354] для задач навигации и вытеснения без агента. Современные объектно-центричные модели на основе слотов с большими сложностями извлекают значимые объектно-центричные представления в визуально сложных средах [252; 471]. В связи с этим тестирования объектно-центричных алгоритмов обучения с подкреплением в визуально насыщенных средах, таких как Habitat [305], становится сложной задачей из-за низкого качества представлений. Однако визуальная простота выбранных для тестирования сред позволяет объектно-центричным моделям извлекать высококачественные представления объектов. Это позволяет сосредоточиться на проблеме объектно-центричного обучения с подкреплением на основе модели, что является основной целью данного раздела диссертационного исследования.

Задача достижения объекта Object Reaching

В задаче Object Reaching фиксированный целевой объект (фиолетовый куб) и набор объектов, являющихся отвлекающими факторами (оранжевый, желтый и голубой кубы), случайным образом размещаются на некоторой сцене. Агент управляет роботом с тремя

актуаторами и должен дотянуться до целевого объекта одним из них (два других постоянно зафиксированы), чтобы получить положительное вознаграждение и решить задачу.

Эпизод заканчивается без вознаграждения, если актуатор касается одного из отвлекающих объектов. Пространство действия в этой среде состоит из трех непрерывных положений сустава подвижного актуатора. В ходе проведенных экспериментов было обнаружено, что один из базовых алгоритмов чувствителен к выбору цветовой схемы для кубов. Поэтому были также проведены эксперименты в задаче с оригинальной цветовой схемой [129]: цвет целевого куба — синий, а цвета отвлекающих кубов — красный, желтый и зеленый.

Задачи навигации Navigation5x5 и Navigation10x10

Среда Shapes2D представляет собой четырехсвязный клеточный мир, где объекты представлены в виде фигур простых форм. Один из объектов, крест, выбран в качестве неподвижной цели. Остальные объекты являются подвижными. Агент управляет всеми подвижными объектами. За один шаг агент может переместить объект в любую свободную соседнюю клетку. Цель агента — столкнуть управляемые объекты с целевым объектом. При столкновении объект исчезает, а агент получает вознаграждение в размере +1.

Когда объект сталкивается с другим подвижным объектом или границами поля, агент получает вознаграждение в размере -0,1, при этом положения объектов не меняются. За каждый шаг в среде агент получает вознаграждение в размере -0,01. Эпизод заканчивается, если на поле остается только целевой объект.

В экспериментах была использована среда размером 5×5 с пятью объектами и среда размером 10×10 с восемью объектами. Пространство действий в среде Shapes2D дискретно и состоит из 16 действий для задачи навигации 5×5 (четыре подвижных объекта) и 28 действий для задачи навигации 10×10 (семь подвижных объектов).

Задача вытеснения без агента PushingNoAgent5x5

Как и в задаче навигации, агент управляет всеми подвижными объектами, но допускаются более сложные взаимодействия между двумя подвижными объектами: при столкновении оба объекта движутся в направлении движения. Агенту необходимо подтолкнуть другой подвижный объект к цели, управляя текущим объектом. Перемещаемый объект исчезает, и агент за такое действие получает вознаграждение в размере +1. Когда контролируемый в данный момент объект сталкивается с целевым объектом или границами поля, агент получает вознаграждение в размере -0,1.

Когда агент выталкивает подвижный объект за границы поля, он получает вознаграждение в размере $-0,1$. За каждый шаг в среде агент получает вознаграждение в размере $-0,01$. Эпизод заканчивается, если на поле остаются только целевой объект и один подвижный объект. В экспериментах была использована среда размером 5×5 с пятью объектами.

Результаты экспериментов

Для задач Navigation 5×5 и PushingNoAgent 5×5 была применена одна и та же модель SLATE, поскольку они используют одно и то же пространство наблюдений. Однако для задачи Navigation 10×10 была обучена отдельная модель SLATE, как и для каждой версии задачи достижения объекта.

В задачах непрерывного достижения объекта в качестве гиперпараметра целевой энтропии для ROCA обычно используется размерность пространства действий. Для двумерных задач с дискретным пространством действий энтропия однородной случайной стратегии масштабируется с помощью настраиваемого адаптивного коэффициента.

Было проведено сравнение ROCA с безмодельным алгоритмом, основанным на PPO, который использует ту же предварительно обученную модель SLATE с зафиксированными параметрами в качестве средства извлечения признаков. Чтобы объединить представления скрытых объектов в единый вектор, подходящий для аппроксиматоров полезности и стратегии PPO, использовался трансформерный кодировщик [137] в качестве объединяющего слоя. В экспериментах в качестве базового алгоритма применялась реализация PPO на основе трансформера, представленная в [129] под названием OCRL. Для задачи достижения объекта были использованы те же значения гиперпараметров, которые были выбраны авторами. Для задач Shapes2D гиперпараметры базового алгоритма OCRL были дополнительно оптимизированы.

Поскольку не существует современных объектно-центрических алгоритмов MBRL, для сравнения в качестве базового MBRL был выбран алгоритм DreamerV3 [421]. Чтобы обеспечить справедливое сравнение между ROCA и DreamerV3, были проведены эксперименты, в ходе которых DreamerV3 использовался с предварительно обученным кодировщиком, полученным из модели DreamerV3, решающей поставленную задачу. Для всех задач были проведены эксперименты с использованием двух различных режимов: один с зафиксированным кодировщиком, а другой с кодировщиком, в котором веса не фиксировались. Однако какого-либо улучшения скорости сходимости не наблюдалось по сравнению с моделью DreamerV3, которая не использует предварительно обученный кодировщик. Кроме того, было обнаружено, что предварительно обученная модель мира значительно ускоряет сходимость DreamerV3, но этот режим делает сравнение несправедливым по отношению к ROCA.

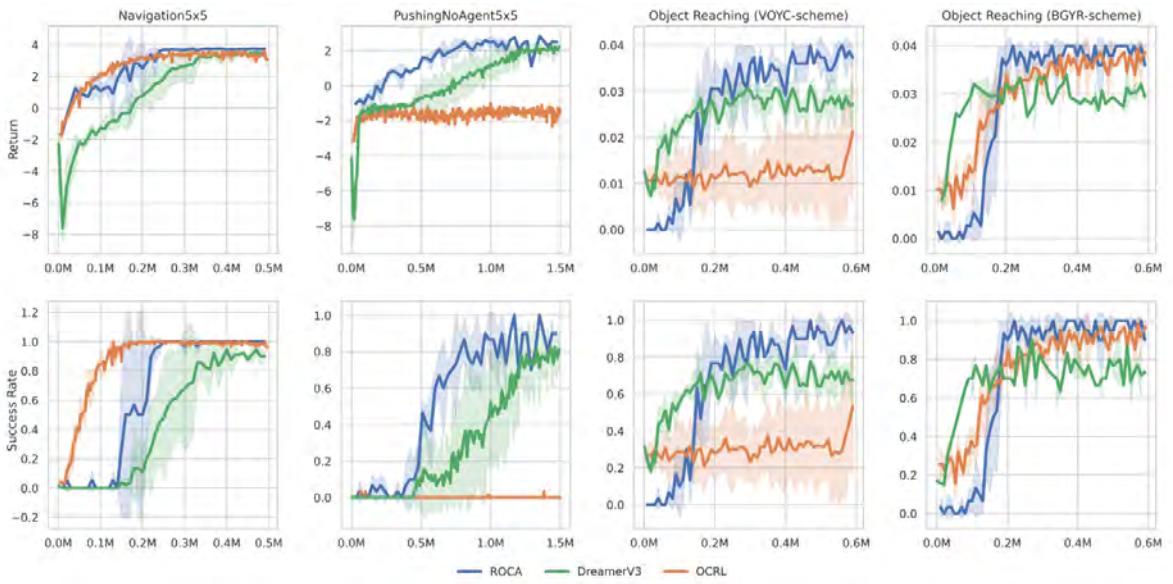


Рисунок 4.13 — Показатель отдачи и успешности решения задачи усредненное по более чем 30 эпизодам и трем запускам эксперимента для моделей ROCA, DreamerV3 и OCRL. ROCA учится быстрее или достигает более высоких показателей, чем базовые алгоритмы.

Заштрихованные области указывают на стандартное отклонение.

Для алгоритма DreamerV3 были использованы значения гиперпараметров по умолчанию из официального репозитория.

Графики на рисунке 4.13 показывают, как отдача по эпизодам у ROCA и у базовых алгоритмов зависит от количества шагов для задач Navigation5x5, PushingNoAgent5x5 и двух версий задачи достижения объекта. Для задачи Navigation5x5 ROCA работает лучше, чем базовый OCRL. Несмотря на то, что DreamerV3 демонстрирует несколько более стабильное и эффективное обучение, чем ROCA, ROCA в конечном итоге обеспечивает более высокую отдачу.

В задаче PushingNoAgent5x5 ROCA превосходит оба базовых алгоритма. Базовые алгоритмы изначально более эффективны в задаче достижения объекта с используемой цветовой схемой, но ROCA превосходит их после 200 тысяч шагов. Для задачи достижения объекта с исходной цветовой схемой базовый алгоритм OCRL демонстрирует гораздо лучшую производительность, но ROCA также превосходит оба базовых подхода после 200 тысяч шагов. Рисунок 4.14 демонстрирует результаты в более сложной задаче Navigation10x10. Оба базовых алгоритма не дают положительного результата. ROCA работает лучше, чем указанные подходы, но не может решить задачу полностью, так как перемещает к цели только пять из семи объектов.

На основе вышеизложенного можно сделать вывод, что низкая производительность базового алгоритма OCRL в задаче достижения объекта с цветовой схемой VOYC обусловлена ее чувствительностью к качеству модели SLATE. Одним из возможных решений для преодоления этой проблемы может служить увеличение количества итераций обучения для SLATE.

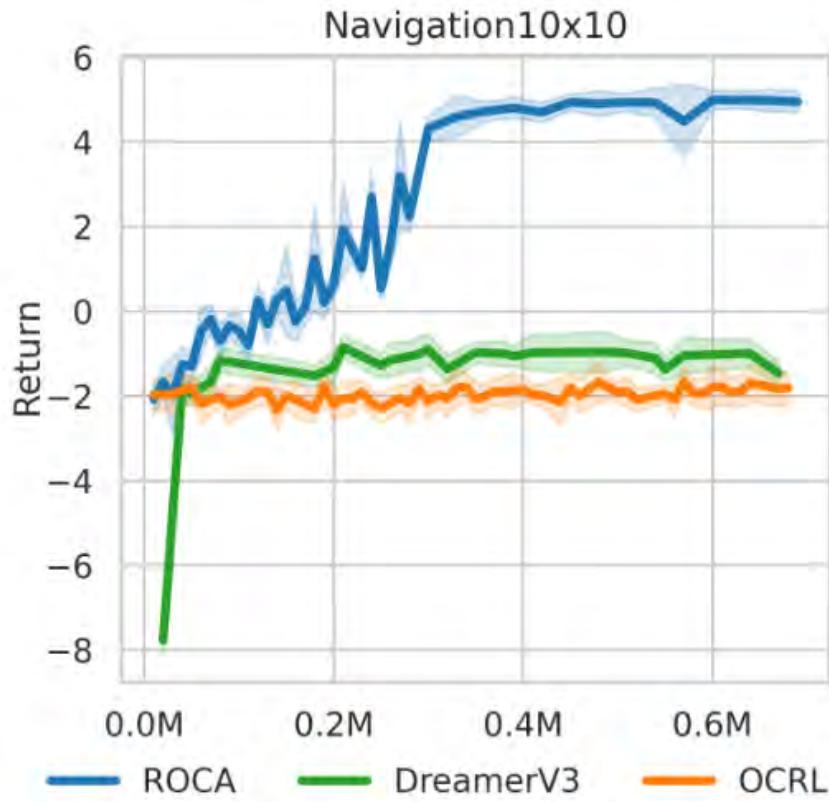


Рисунок 4.14 — Среднее значение отдачи, усредненное по 30 эпизодам и трем запускам эксперимента, для моделей ROCA, DreamerV3 и OCRL в задаче Navigation10x10. ROCA демонстрирует лучшую производительность, чем базовые алгоритмы, но все равно не решает задачу. Заштрихованные области указывают на стандартное отклонение.

Покомпонентный анализ

ROCA построена на базе алгоритма SAC, и, таким образом, исследование влияния различных компонент должно быть направлено на оценку влияния различных модификаций, которые были внесены в исходный SAC с монолитным сверточным кодировщиком на базе CNN. Рисунок 4.15 иллюстрирует результаты дополнительных экспериментов по оценке эффектов предварительно обученного кодировщика SLATE, объектно-центричного актора и критика, объектно-центричной модели мира и целевой настройки энтропии. Была проведена оценка качества нескольких монолитных и объектно-центрических версий SAC и проведено их сравнение с ROCA.

SAC-CNN — это стандартная монолитная версия SAC, которая использует сверточный кодировщик из оригинальной реализации DQN [323]. В SAC-SLATE сверточный кодировщик заменен на предварительно обученный кодировщик SLATE с зафиксированными параметрами, в то время как другие компоненты модели остаются прежними. Чтобы получить монолитное представление состояния z_t^* из объектно-центрического z_t , формируемого SLATE, было взято среднее значение по объектной размерности:

$z_t^* = \sum_{i=0}^K z_t^i / K$. Необходимо обратить внимание, что z_t^* не зависит от порядка слотов в z_t и может быть передан в стандартные персептроны актора и критика.

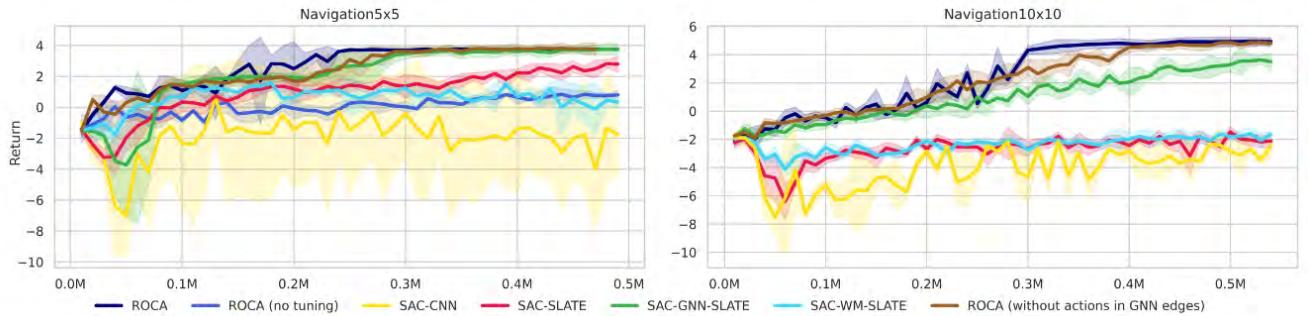


Рисунок 4.15 — Исследование влияния компонент. SAC-CNN — версия SAC со стандартным сверточным кодировщиком. SAC-SLATE — версия SAC с предварительно обученным кодировщиком, который усредняет представления объектов для получения итогового представления текущего состояния. SAC-WM-SLATE — модификация SAC-SLATE, которая использует монолитную модель мира в своем критике. SAC-GNN-SLATE — объектно-центричная версия SAC с предварительно обученным кодировщиком, который использует GNN в акторе и критике. ROCA (без настройки) — версия ROCA без настройки целевой энтропии. ROCA (без действий в ребрах GNN) — версия ROCA, в которой функции ребер не принимают действия в качестве входных данных. ROCA превосходит рассмотренные базовые алгоритмы. Заштрихованные области указывают на стандартное отклонение.

SAC-WM-SLATE основан на SAC-SLATE и может рассматриваться как монолитная версия ROCA. В нем модели актора, полезности состояния, вознаграждения и переходов реализованы с использованием обычных перспекtronов. SAC-GNN-SLATE является объектно-центричной версией SAC и может рассматриваться как ROCA без модели мира в модуле критика. В ней используется предварительно обученный кодировщик SLATE с зафиксированными параметрами и модули актора и критика на основе GNN.

Кроме того, было проведено сравнение ROCA с вариантом, в котором целевая энтропия установлена на значении по умолчанию, равном масштабированной энтропии равномерного распределенной стратегии с коэффициентом 0,98 [176].

Такой покомпонентный анализ показал, что в монолитном режиме модель SLATE значительно улучшает производительность только в относительно простой задаче Navigation5x5. Однако расширение критика с помощью модели мира не улучшает скорость сходимости. Объектно-центричный SAC-GNN-SLATE превосходит все монолитные модели. Наконец, ROCA, который использует объектно-центричную модель мира в модуле критика, превосходит SAC-GNN-SLATE. Также необходимо обратить внимание, что представленные результаты были получены после точной настройки гиперпараметров для всех моделей.

В данном разделе диссертационного исследования представлен алгоритм ROCA, объектно-центричный метод обучения с подкреплением на основе отложенного опыта и модели полезности, который использует предварительно обученную модель SLATE в качестве объектно-центричного подхода к извлечению признаков. Проведенные

эксперименты с 3D- и 2D-задачами демонстрируют, что ROCA формирует эффективные стратегии и превосходит объектно-центричные базовые безмодельные алгоритмы и алгоритмы, использующие модель среды.

Модель мира, построенная на архитектуре GNN, демонстрирует, что графовые нейронные сети могут быть успешно применены в задачах MBRL для обучения стратегии. Несмотря на то что была использована модель SLATE в качестве объектно-центричного подхода к извлечению объектов, в принципе, можно заменить SLATE другими объектно-центричными моделями на основе слотов.

Впрочем, у представленного алгоритма ROCA есть и ограничения. Во-первых, его модель мира детерминирована, и имеются сложности по предсказанию динамики в сильно стохастических средах. Кроме того, поскольку модель ROCA основана на алгоритме SAC, она чувствительна к выбору гиперпараметра целевой энтропии, особенно в средах с дискретными пространствами действий [530; 605].

В направлении развития предложенного алгоритма обучения стратегии на основе объектно-центричных представлений в контексте архитектуры NSLP возможны расширения ROCA для работы в более сложных с визуальной точки зрения условиях. Для достижения этого планируется заменить SLATE недавно предложенной моделью DINOSAUR [161], которая показала многообещающие результаты на реалистичных наборах данных. Кроме того, возможно проведение экспериментов с подходами, не ориентированными на слотовые представления, например, с работой [203].

Также к перспективным направлениям относится повышение устойчивости модели к изменениям целевой энтропии путем применения подхода, основанного на метаградиентах [645], который устраняет необходимость в использовании этого гиперпараметра.

Во многих средах одновременно взаимодействует только небольшое количество объектов. Следовательно, представление состояния среды в виде полного графа приводит к избыточным связям. Чтобы решить эту проблему, возможно также внедрить подходы [464; 654], которые разрежают граф состояний.

4.4 Языковые модели в качестве объектно-центричных моделей мира

В данном разделе диссертационного исследования рассмотрен основной цикл работы архитектуры NSLP с объектно-центричным кодированием сцены в виде графового представления, составлением плана действий на основе языковой модели и использованием библиотеки навыков для выполнения получаемой высокоуровневой стратегии (рисунок 4.16). Исследованы особенности декомпозиции задачи при работе языковой модели L , а также

особенности вызова предобученных на основе обучения с подкреплением навыков κ_i . Основные результаты, изложенные в данном разделе, представлены в публикациях [8].

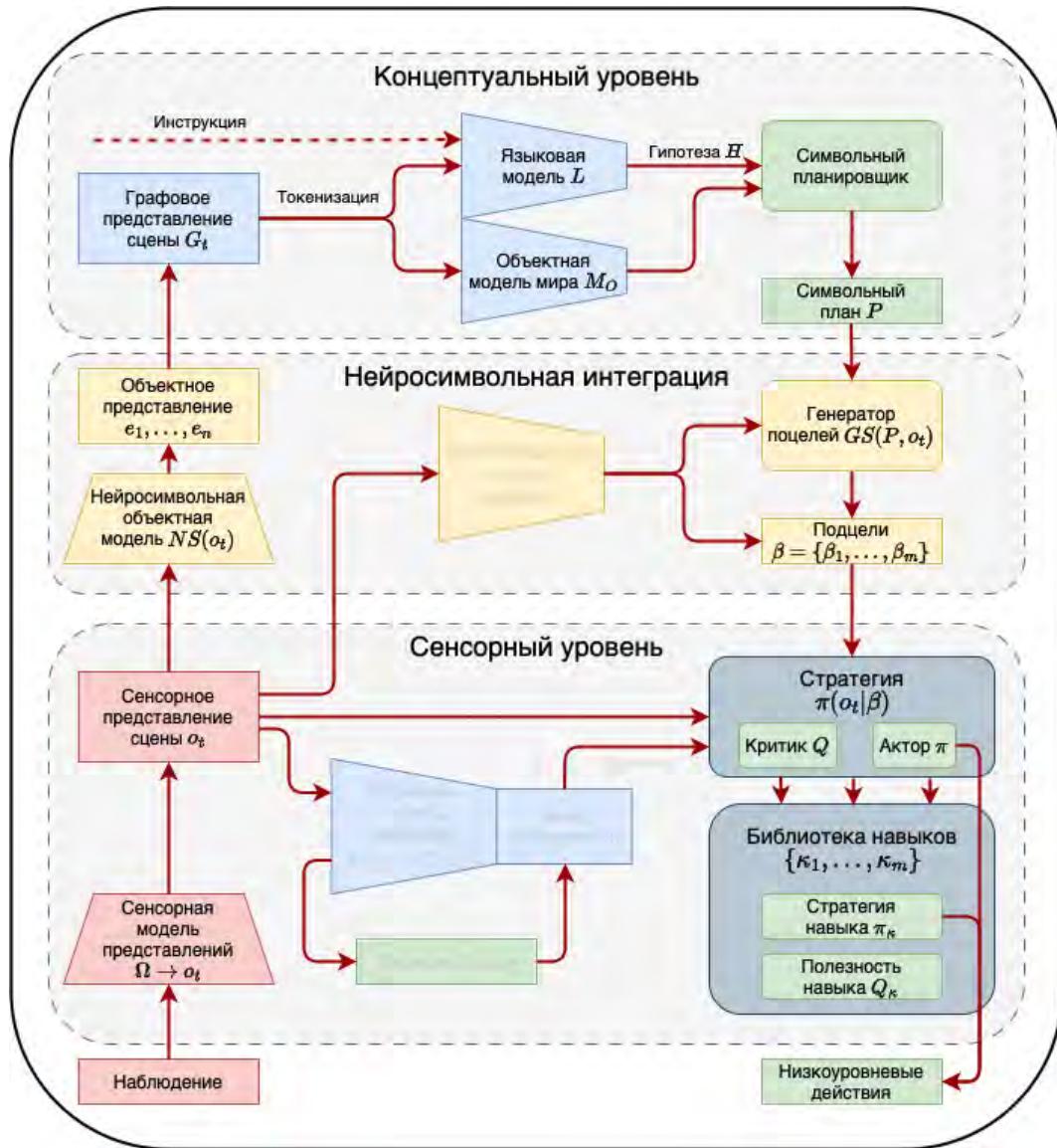


Рисунок 4.16 — Архитектура NSLP с выделенным основным циклом генерации плана действий с использованием языковой модели и их выполнения на основе библиотеки предобученных навыков.

4.4.1 Интеграция языковых моделей и обучения с подкреплением в виртуальных средах

Способность решать сложные задачи, сформулированные на естественном языке и требующие выполнения многошаговой последовательности действий в окружающей среде, является фундаментальным свойством человеческого интеллекта. Недавние значительные успехи больших языковых моделей (БЯМ) в выполнении задачи следования инструкциям

и генерации объяснений демонстрирует их мощные возможности в решении задач общего назначения с использованием общих знаний и в задачах генерации кода по языковым инструкциям (см. раздел 2.4). Однако вероятность успешного выполнения многошаговых задач автономными агентами, управляемыми БЯМ общего назначения, по-прежнему низка [120]. Более того, БЯМ обучаются исключительно на текстовых данных, что ограничивает их способность понимать и выполнять действия в реалистичной симуляционной среде. В связи с этим, даже такие модели как ChatGPT [622] демонстрируют низкое качество в задачах пространственных рассуждений [106]. С другой стороны, обучение с подкреплением доказало свою эффективность при формировании последовательностей детализированных действий для конкретных задач в среде. Таким образом, изучение способов интеграции БЯМ, направленных на понимание естественного языка и планирование на высоком уровне, с методами обучения с подкреплением, направленных на обучение стратегии по манипуляции объектами в окружающей среде, представляет собой многообещающее направление исследований.

БЯМ можно рассматривать как универсальные базы знаний, которые позволяют пользователям взаимодействовать с ними на естественном языке и решать сложные задачи [378; 578]. Недавние исследования показали, что предварительно обученные БЯМ могут составлять планы действий высокого уровня как в симуляционной, так и в реальной средах [144; 262; 457]. Однако эти, основанные на БЯМ подходы требуют ручного задания формата запросов, основанного на правилах перевода языковых команд в действия воплощенного агента, и реализуют стратегии выбора действий с учетом цели только из распределения потенциальных вариантов, генерируемых языковой моделью. В этом контексте несколько недавних исследований [233; 622] продемонстрировали, что верхнеуровневые планы действий, извлеченные из языковых моделей, могут быть уточнены с помощью обучения с подкреплением.

В данном разделе диссертационного исследования предлагается новый подход, включающий этап конструирование подзадач по языковой инструкции (*Language to Subtask Builder, L2S*), для обучения агентов в мультимодальных лингво-визуальных средах. Этот метод основан на раздельном обучении языковой модели и стратегии агента. В описываемой архитектуре предлагается рассматривать задачу обучения стратегии агента как целенаправленную задачу, что обеспечивает лучшее качество обобщения при решении разнообразных задач в симуляционной среде. Кроме того, представлено несколько подходов к обучению стратегии в случае ограниченного набора данных. Некоторые из этих подходов основаны на увеличении объема данных с использованием модели ChatGPT, в то время как другие полагаются на разделение подзадач путем изменения формата представления данных и разбивки используемых подзадач на примитивы.

Эффективность разрабатываемого подхода была протестирована в упрощенной среде для обучения воплощенных агентов IGLU [325], где агент «Builder» создает структуры на основе инструкций на естественном языке, получаемых от агента «Architect» (см. пример на

рисунке 4.17). Полученные результаты показывают, что метод L2S превосходит алгоритмы, представленные на конкурсе NeurIPS IGLU⁵ [331].

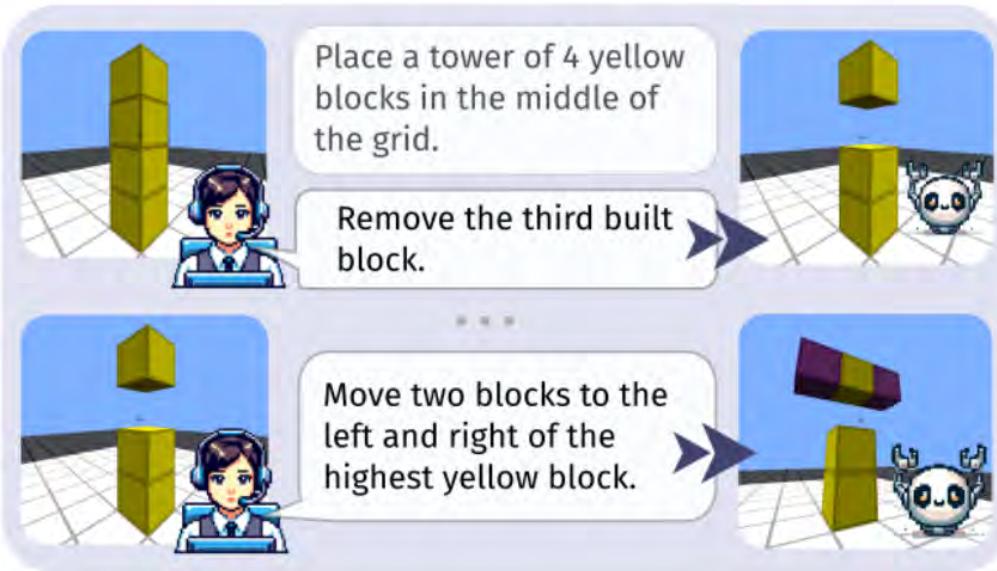


Рисунок 4.17 — Задача коллаборативного взаимодействия в среде агента и пользователя включает в себя следующие этапы: пользователь предоставляет инструкции агенту, а агент создает фигуру в среде на основе этих инструкций.

В недавних исследованиях активно исследуется возможность создания планов действий с использованием языковых моделей. Некоторые работы сосредотачиваются на проблеме составления запросов для эффективной генерации плана [511], в то время как другие решают проблему перевода выходных данных языковой модели в исполняемые действия [378]. В исследовании [233] используются модели, построенные с помощью метода обучения с подкреплением, для выбора возможных действий и выполнения элементарных действий. Многие из имющихся в данное время работ в этой области [124; 332] направлены на интерактивное управление робототехническими системами в режиме реального времени. В целом, необходимо отметить, что задача следования языковым инструкциям является важной областью исследований в воплощенном искусственном интеллекте последнего времени [225; 606].

Разработка эффективного процесса обучения и предоставление достаточного количества обучающих данных являются основными проблемами при создании интерактивных воплощенных агентов, которые обучаются решать задачи, описанные инструкциями на естественном языке, соотнесенные с условиями в коллаборативной среде. В работе [458] авторы представляют среду, разработанную на основе среды Minecraft, и набор данных, содержащий примеры взаимодействия эксперта-архитектора и агента-строителя. Архитектор знает структуру или форму, которая должна быть построена в среде, и должен объяснить агенту-строителю, как ее построить, используя инструкции на естественном языке. В упомянутой работе в первую очередь рассматривается вопрос разработки и оценки

⁵<https://www.iglu-contest.net/>

модели для прогнозирования инструкций архитектора. Данное направление исследований было продолжено в другой работе [563], в которой все высказывания агента-строителя, включая уточняющие вопросы, разделены на восемь типов. Основываясь на аннотированных данных, предлагаемая в этой работе модель агента-строителя может определять, когда следует запросить разъяснения или выполнить конкретные инструкции. Организаторы конкурса IGLU [331; 8] расширили этот набор данных и предоставили базовое решение задачи построения фигур в виртуальной среде. Этот базовый метод использует вектор признаков окружающей среды в качестве наблюдения, в том числе положение агента и направления его движения, а также положения блоков, составляющих целевую фигуру.

В работе [163] предлагается подход по интерпретации инструкции на естественном языке для преобразования команд, представленных в произвольной форме, в исполняемые действия, состоящий из двух этапов: использование парсера на основе представлений абстрактных понятий (AMR) и дальнейшее преобразование результата в диалоговую форму. Задача прогнозирования последовательности действий агента-строителя, в которой применяются только действия по размещению и удалению блоков (без конкретного воплощения самого агента), рассматривается в работе [337]. Аналогичная задача была решена в еще одном исследовании [395], в котором предлагается новая архитектура для одновременного обучения архитектора и строителя в простейшей клеточной среде.

Метод совместного решения проблем под управлением эксперта-человека [321; 379] предполагает реализацию непараметрического подхода, который использует примитивы на основе правил и планировщик иерархической сети задач (HTN) для построения структур. Использование планировщиков HTN для планирования действий по решению задачи не является новой концепцией, поскольку он уже использовался в предыдущей работе [276], которая предлагала выделять в пространстве целей агента несколько уровней абстракции. В этом случае целями были подзадачи по построению определенных частей более крупной фигуры. Напротив, предлагаемый в данном разделе подход предполагает тонкую настройку языковой модели, декомпозицию общей задачи на подзадачи независимо от последовательности их выполнения и использование предварительно обученных навыков навигации в среде на основе обучения с подкреплением, которые адаптируют высокоуровневый план из подзадач к выполнению в окружающей среде.

4.4.2 Объектная декомпозиция плана действий в среде IGLU

Среда IGLU

Для рассматриваемой задачи выполнения инструкции воплощенным агентом была использована среда IGLU⁶, где агент может создавать пространственные структуры из блоков (фигур) разных цветов. Цель агента — выполнить задачу, выраженную в виде инструкции, представленной на естественном языке.

Пространство наблюдения состоит из изображения от первого лица (POV) размером (64, 64, 3), количество предметов инвентаря составляет 6, а углы тангажа и рыскания имеют значение 5. Агент может перемещаться по зоне, доступной для размещения блоков, устанавливать и удалять блоки, а также переключаться между различными типами блоков. Кроме того, среда предоставляет текстовое описание наблюдения в виде диалога из набора данных, который определяет задачу строительства.

Целевые высказывания определяют оставшиеся блоки, которые необходимо добавить. Симуляционная среда предоставляет два режима действий по перемещению на выбор: ходьба и полет. В настоящей работе в среде использовался режим полета, в котором, в отличие от ходьбы, нет учета эффектов гравитации, но действия по изменению положения непрерывны. В целом, пространство действий объединяет дискретные и непрерывные подпространства: дискретное пространство из 13 действий (поор, четыре действия поворота камеры, убрать блок, разместить блок, выбрать тип блока 1–6) и набор из шести непрерывных действий для перемещения во всех трех направлениях.

Вознаграждение зависит от того, насколько точно построенная фигура соответствует цели, но не зависит от того, где была построена фигура в пространстве. Для подсчета метрики запускается процедура, реализованная с помощью функции свертки и вычисляющая некоторое расстояние между имеющимся клеточным состоянием и целевым. Рассчитывается побочное пересечение для каждого варианта смещения фигуры, а затем оно максимизируется по всем возможным смещениям.

В проведенном исследовании для оценки качества разрабатываемого подхода была использована метрика $F1$, предложенная в исходном конкурсе IGLU, и ее модификация *Strict F1*. Первая метрика, $F1$, ориентируется на определение максимального пересечения трехмерных вокселей между исходной структурой и структурой, построенной агентом. Вторая метрика, *Strict F1*, налагает штраф на структуру, если процесс ее построения отклоняется от последовательности, описанной в диалоге из набора данных.

Оказывается, что прямое перекрытие, измеряемое с помощью метрики $F1$, может не учитывать визуально значимую информацию. Чтобы устранить этот недостаток необходимо учитывать не только абсолютное положение каждого блока, но и их взаимосвязь внутри

⁶<https://github.com/iglu-contest/gridworld>

структуры. Следовательно, если некоторые блоки правильно расположены относительно друг друга, но имеют незначительные ошибки в абсолютном позиционировании, структура будет оценена выше по сравнению с F1, поскольку она учитывает визуальную близость структуры к необходимой в целом, а не только в плане ее элементов.

Чтобы лучше отразить структурное сходство, которое и опосредует визуальное сходство, была разработана метрика под названием *воксельное структурное сходство (VSS)*. VSS количественно определяет пространственные отношения между блоками, кодируя их в терминах угловых расстояний до соседних блоков, которые затем сортируются на основе их близости к целевому блоку. При вычислении этого показателя объединяются блоки из исходной и прогнозируемой структур с присвоением каждой паре оценки сходства. Наилучшие совпадения определяются с использованием венгерского алгоритма, и затем их оценки сходства усредняются.

В ходе проведенных исследований оказалось, что оптимальная система оценки должна охватывать все три показателя для проведения всесторонней оценки эффективности работы агента. Перечисленные показатели агрегируются с помощью *гармонического среднего (HM)*, которое характеризуется своей чувствительностью к низким значениям и гарантирует, что более высокий балл может быть достигнут только за счет высоких значений по всем показателям.

Для сравнения были выбраны три лучших решения из конкурса IGLU 2022. Алгоритмы T5 [256] и Pegasus [492], занявшие третье и второе места соответственно, основаны на предложенном организаторами решении, но отличаются используемыми языковыми моделями. Было также включено решение команды, занявшей первое место, под названием BrainAgent⁷, чей подход значительно отличается от других. Авторы этого решения использовали сквозное обучение, при этом функция стратегии использовала в качестве входных данных представление, полученное из языковой модели с фиксированными весами, а также информацию об окружающей среде и некоторые признаки, добавленные вручную.

Общая процедура обучения алгоритма L2S

Архитектура метода L2S разделена на два отдельных обучаемых компонента: языковой модуль и модуль стратегии выполнения действий. Языковой модуль отвечает за реализацию ключевой функции преобразования предоставляемых пользователем языковых команд в отдельные подзадачи. После этого модуль стратегии отвечает за реализацию этих подзадач в используемой виртуальной среде.

Ключевым дополнением к этим компонентам является модуль менеджера задач (Task Manager). Этот компонент выступает в качестве опосредующего модуля, обеспечивающего

⁷<https://github.com/kakaobrain/brain-agent>

плавную интеграцию и координацию между языковым модулем и модулем стратегии, тем самым повышая общую эффективность и функциональность общего алгоритма L2S.

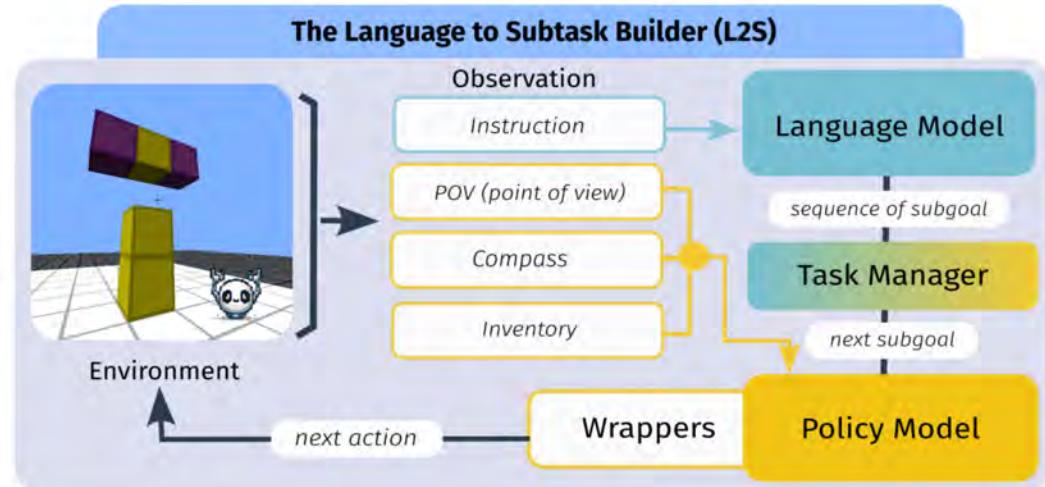


Рисунок 4.18 — Схема предлагаемого метода построения подзадач по языковой инструкции (L2S). Агент состоит из трех частей. Сначала языковой модуль преобразует инструкции на естественном языке в текстовую последовательность подзадач. Затем менеджер задач преобразует эту последовательность в упорядоченный список подзадач. Наконец, модуль стратегии выполняет предварительно обученную стратегию для выполнения каждой подцели, одновременно отслеживая успешность действий.

Предложенная декомпозиция решения задачи (см. рисунок 4.18) повышает эффективность обучения с подкреплением, позволяя генерировать различные сценарии подзадач во время обучения, что в свою очередь помогает охватить более широкий спектр потенциально возможных состояний среды. Кроме того, разделение этих модулей позволяет гибко модифицировать выходные данные языковой модели, тем самым расширяя область применения и открывая новые возможности для экспериментов.

Языковой модуль в алгоритме L2S может качественно справляться с обработкой языковых инструкций и преобразованием их в последовательность подзадач. В частности, в среде IGLU эти подзадачи были определены как задачи размещения или удаления отдельного блока в указанном месте или удаления и размещения простой прямоугольной формы.

Для выполнения задач по формированию конкретного плана действий на основе предоставленного текстового описания наблюдения была выбрана модель типа кодировщик-декодировщик Flan-T5⁸. Ключевым фактором при выборе этой модели было то, что модель Flan-T5 была предварительно обучена с использованием инструктивных наборов данных[552].

Модуль менеджера задач действует как компонент преобразования задачи, сгенерированной языковым модулем, в формат, обрабатываемый модулем стратегии. Если среда позволяет разложить подзадачу на дополнительные шаги в соответствии с конкретными правилами, продиктованными средой, эту роль также выполняет менеджер задач. Кроме того, этот модуль реализует определенную логику в последовательной

⁸<https://huggingface.co/google/flan-t5-base>

генерации целей для модуля стратегии. Эта логика в генерации целей может быть либо получена из языковой модели, либо предопределена на основе конкретных правил.

Например, в случае с некоторой фигурой в среде IGLU, этот модуль выполняет следующие задачи: он преобразует текстовую последовательность подзадач в трехмерный воксель. В сценариях, где необходимо создавать блоки в воздухе, он генерирует вспомогательные блоки, которые потом удаляются. Как правило, для генерации последовательности подзадач используется подход «снизу вверх, слева направо».

В проведенном исследовании основной целью являлось решение подзадач в среде, которая предполагает последовательное принятие решений. Это в точности соответствует постановке задачи для обучения с подкреплением, особенно с учетом того, что задача использует визуальные высокоразмерные наблюдения для формирования решений. В связи с этим необходимо предложить такую реализацию модуля стратегии, которая бы поддерживала как размещение, так и удаление определенных блоков в среде. Чтобы сузить пространство действий и сконцентрироваться на навигации агента к целевому местоположению, было проведено предобучение этого модуля именно на задаче навигации.

Для обеспечения эффективного выполнения действий по размещению или удалению блока после завершения подзадачи навигации была использована специальная надстройка над стратегией агента. Эта надстройка определяет, должен ли агент размещать или удалять блок в целевом местоположении, и активируется, когда агент инициирует действие *готово* в соответствии с конкретными требованиями выполнения подзадачи.

Для реализации этого модуля была использована обычная архитектура «актор-критик» с аппроксиматором стратегии в виде сверточного кодировщика входных данных с остаточными слоями, схожего с архитектурой аппроксиматора в алгоритме IMPALA [326], которая часто используется для обработки RGB-изображений. Однако в реализованной версии этой архитектуры были опущены слои объединения по максимуму (max-pooling) при кодировании целевого блока. Выходные данные этих кодировщиков затем объединяются с дополнительной информацией о среде (направление движения и состав инвентаря) и затем передаются в слой многослойного персептрона (MLP). Этот результирующий вектор затем передается в слой долговременной кратковременной памяти (LSTM), как это обычно реализуется в рекуррентных нейронных сетях [319].

В качестве базового алгоритма собственно обучения была использована быстрая высокопроизводительная реализация метода оптимизации ближайшей стратегии (РРО), предложенная в работах [544] и в [512].

Формирование обучающей выборки

Обучающий и тестовый наборы данных содержат 109 и 41 инструкцию на английском языке с соответствующими воксельными представлениями, которые необходимо построить⁹. Каждая инструкция состоит из отдельных этапов построения фигуры с соответствующим воксельным представлением для каждого из них. Воксельное представление преобразуется в последовательность координат и цветов блоков в виде строки кортежей. Если после выполнения шага инструкции некоторые блоки были удалены, в кортежах для этих блоков вместо цвета будет команда *удалить*. Порядок блоков в последовательности соответствует внешнему виду блоков в клеточной среде со следующими знаками направлений: сначала вдоль оси *x*, затем вдоль оси *y* и, наконец, вдоль оси *z*. Во многих инструкциях не указывается положение первого блока, и он может появляться в любом месте клеточной среды по осям *x* и *y* случайнym образом. Это является дополнительной сложностью для модели во время обучения, поскольку приводит к высоким значениям функции потерь для токенов, которые невозможно определить по входным данным. В связи с этим данная последовательность меняется таким образом, чтобы первый блок первого шага инструкции был помещен в точку начала координат, а остальные блоки были соответствующим образом сдвинуты.

Ограниченный размер и неточный характер исходного набора данных ограничивают производительность языковой модели. Чтобы преодолеть эффекты переобучение и повысить репрезентативность выборки данных, были использованы три стратегии для улучшения качества модели.

Аугментация по цвету. Для каждого элемента в наборе данных были сгенерированы еще два дополнительных элемента с той же инструкцией и целью, но с блоками разных цветов. Это делается путем простой замены названий цветов в инструкции названиями других цветов и соответствующего изменения цветов блоков в целевом наборе вокселей. Это дополнение применяется перед обучением ко всем наборам данных, включая наборы данных, созданные с помощью следующих методов аугментации.

Перефразирование текстовых инструкций. Чтобы улучшить понимание языковой моделью различных способов, которыми пользователи описывают формы, была использована модель ChatGPT для перефразирования исходных текстовых инструкций. Такой подход обеспечивает обучаемую модель дополнительными вариациями одних и тех же инструкций и помогает ей научиться распознавать шаблоны и понимать сложные инструкции.

Обновление координат. Чтобы улучшить понимание моделью концепции сторон света, фигуры были перемещены вдоль разных осей и с помощью модели ChatGPT были сгенерированы новые инструкции для каждой фигуры с использованием обновленной системы координат. Такой подход помогает модели научиться лучше обобщать различные ориентации одних и тех же фигур.

⁹<https://github.com/microsoft/iglu-datasets>

Кроме перечисленных выше процедур аугментации, был предложен метод разложения фигур на примитивы, которые необходимо сконструировать в среде. При более подробном рассмотрении используемого набора данных можно заметить, что каждая отдельная инструкция предполагает построение либо примитива (т.е. параллелепипеда заданных размеров), либо удаление определенных блоков из фигуры, либо размещение блоков поверх существующих.

Таким образом, вместо генерации неограниченной последовательности блоков для каждой инструкции можно предсказать список из трех кортежей. Первый кортеж определяет местоположение, в котором должен быть размещен примитив (ближе всего к началу координат фигуры), второй кортеж определяет его размеры (ширина, длина и высота), а третий дает условное представление цвета примитива.

Таким образом, для таких типичных инструкций, как «*Обращаясь лицом на север постройте плоский квадрат из оранжевых блоков размером 4 на 4 блока в середине среды*», предсказание будет представлять собой не длинный список из 16 координат, а один короткий список вида: `[(0, 5, 5), (0, 4, 4), «оранжевый»]`.

Языковая модель обучается в последовательном режиме (seq2seq) с преобразованием инструкции в последовательность блоков в описанном выше формате с размером пакета данных 8, скоростью обучения 0,00001 с 80000 шагами обучения. Обучение на графическом процессоре Titan RTX заняло 12 часов.

Обучение с подкреплением для агента было проведено в целенаправленном режиме с акцентом на создание распределения потенциальных подзадач. Этот процесс включает в себя разделение построения одной фигуры на несколько отдельных эпизодов, каждый из которых требует от агента достижения определенной промежуточной цели. Как уже было сказано ранее, основной задачей агента является навигация в пространстве среды, следовательно, в каждом эпизоде его цель состоит в том, чтобы переместиться в определенную точку, где он должен разместить или удалить целевой блок.

Учитывая, что при выполнении действий агент часто сталкивается с необходимостью достраивать некоторые фигуры к уже имеющимся блоками (т.е. полностью чистое поле встречается только в начале построения фигуры), в данном процессе обучения особое внимание уделяется инициализации таких частично заполненных сред.

Кроме того, чтобы сделать обучение более ориентированным на построение конкретных фигур из используемого набора данных, была повышена некоторая вероятность того, что для построения будут выбраны фигуры непосредственно из обучающего набора данных.

Функция вознаграждения для агента в процессе обучения определяется на основе расстояния Манхэттена между размещенным блоком и целевой позицией, повторяя функцию вознаграждения базовой реализации IGLU.

Стратегия агента была обучена на 175 миллионах шагов в среде, это заняло 6 часов на графическом процессоре Titan RTX.

4.4.3 Экспериментальное исследование объектной декомпозиции

Результаты обучения модуля стратегии

В соревновании IGLU обучение базового алгоритма приводит к тому, что полученный агент располагает блоки непосредственно под собой. Несмотря на вытекающую из этого большую предсказуемость поведения, это ограничивает формирование сложных стратегий, таких как построение соседних блоков для завершения фигуры.

Для повышения производительности такой стратегии было оптимизировано пространство действий агента за счет исключения действия по удалению и добавлению блоков. Эти действия были перенесены в языковую модель и менеджер задач. При изменении пространства действий среды в режиме полета агент был освобожден от необходимости создавать вспомогательные блоки для построения других блоков. Это изменение не только свело к минимуму вероятность ошибок при построении, но и сократило количество действий, необходимых для завершения фигуры.

В результате указанных модификаций усовершенствованная стратегия агента значительно превзошла базовый уровень, достигнув показателя успешности выполнения задачи (SR) 0,95 по сравнению с базовым показателем 0,85.

Сравнение процедур аугментации данных

Было проведено сравнение обучения языковой модели для различных процедур аугментации данных (табл. 14), включая увеличение исходного набора данных (процедура A), аугментацию цвета и источника света (процедура ABC) и изменение формата набора данных на использование примитивов (процедура prim). Также была обучена модель, использующая стратифицированные подмножества (A (strat) и ABC (strat)), выбранные из исходного и дополненного наборов данных.

Первоначальные эксперименты с простой аугментацией набора данных не привели к существенному улучшению, что привело к тому, что были рассмотрены дополнительные методы аугментации. Было обнаружено, что обучение на дополненных данных помогает модели научиться распознавать шаблоны, а использование цветов улучшает построение шаблона.

Однако было также замечено, что аугментация по цвету сама по себе не помогает определить основное направление построения фигуры. Оказывается, что стратификация с использованием данных как с аугментацией по цвету, так и с увеличением доли данных с изменением направления (методы ABC strat) решает эту проблему.

Таблица 14 — Сравнение метрики F1 базовой модели FlanT5 при обучении с использованием двух разных методов аугментации данных. Метод A предполагает замену цветов, в то время как метод ABC включает как замену цветов, так и вращение фигур. Наборы данных A(strt) и ABC(strt) были использованы для выборки эпох обучения путем стратификации. В столбце «data» представлены результаты для модели, обученной на исходном наборе данных IGLU, в то время как в столбце «prim» показаны результаты для модели, обученной на исходном наборе данных, где каждая инструкция преобразуется с использованием метода декомпозиции фигуры на примитивы.

	data	A	ABC	A(strt)	ABC(strt)	prim
VSS	0.24	0.28	0.24	0.26	0.27	0.32
F1	0.42	0.45	0.45	0.43	0.48	0.5
Strict F1	0.35	0.38	0.38	0.35	0.42	0.43
HM	0.32	0.36	0.33	0.33	0.37	0.4

Метод аугментации, включающий использование примитивов (prim), обеспечивает заметный прирост по всем показателям без дополнительных аугментаций. Из проведенных экспериментов следует, что лучшим методом расширения набора данных является метод с обучением на выделенных примитивах. Вследствие такого подхода были получены формы, которые структурно намного ближе к целевым в соответствии с метрикой VSS, и результаты достигаются даже без добавления дополнительных данных. Однако значимым недостатком этого метода является высокая среднеквадратичная ошибка для позиций ($MSE = 3,5$), указывающая на то, что основной проблемой, по-видимому, является определение положения примитивов в клеточной среде.

Результаты на наборе данных IGLU

В рамках проведенных экспериментов были исследованы возможности использования анализа нескольких примеров (few-shot) и специальных запросов к модели ChatGPT для прогнозирования координат в среде. Были сформулированы запросы, описывающие среду, в которой ChatGPT требуется создать определенные фигуры. Это описание включает определение системы координат, объяснение значений приращений координат по отношению к сторонам света и словарь типичных терминов, связанных с пространственными свойствами среды (например, строка, столбец, башня и т.п.), которые определяют положение примитивов фигур. За один сеанс обращения к ChatGPT передавались последовательные инструкции для одной фигуры. Задачей модели было распознать и выполнить указанные действия (см. табл. 15).

Таблица 15 — Средние значения метрик по всем тестовым заданиям в наборе данных IGLU по пяти запускам экспериментов. Алгоритмы T5 (*770 миллионов параметров*) и Pegasus (*568 миллионов параметров*) — это реализации агентов из общедоступного решения соревнования IGLU. BrainAgent — это сквозная модель с 640 миллионами параметров без декомпозиции метода на языковую модель и модель стратегии. L2S работает с *220 миллионами параметров* на базе Flan-T5. Лучший подход выделен **полужирным**, а лучшая модель, включая проприетарные реализации, такие как Chat-GPT, обозначена розовым цветом

ПРЕДСКАЗАНИЕ БЛОКОВ ПО ТЕКСТУ

	T5 (MHB)	Pegasus	BrainAgent	L2S (coords)	L2S (prim)	L2S (GPT)	L2S (GPT prim)
NLP VSS	0.14	0.14	-	0.27	0.32	0.31	0.26
F1	0.37	0.36	-	0.48	0.5	0.63	0.34
Strict F1	0.32	0.31	-	0.42	0.43	0.45	0.20
HM	0.23	0.23	-	0.37	0.4	0.43	0.25

СРЕДА IGLU

	T5 (MHB)	Pegasus	BrainAgent	L2S (coords)	L2S (prim)	L2S (GPT)	L2S (GPT prim)
VSS	0.05 ± 0.04	0.05 ± 0.05	0.15 ± 0.05	0.23 ± 0.02	0.25 ± 0.01	0.33 ± 0.04	0.19 ± 0.05
F1	0.07 ± 0.07	0.07 ± 0.07	0.35 ± 0.09	0.39 ± 0.03	0.43 ± 0.03	0.49 ± 0.07	0.25 ± 0.08
Strict F1	0.06 ± 0.07	0.06 ± 0.06	0.21 ± 0.07	0.30 ± 0.03	0.39 ± 0.03	0.42 ± 0.06	0.21 ± 0.08
HM	0.06	0.06	0.21	0.3	0.34	0.4	0.21

В проведенном исследовании разработанный метод L2S оценивался по сравнению с решениями, представленными на конкурсе IGLU 2022. Как показано в табл. 15, предлагаемое решение превосходит сравниваемые алгоритмы по всем показателям. В случае с метрикой VSS видно, что в соответствии с ней алгоритм L2S создает фигуры с большим структурным сходством. Это указывает на то, что он правильно размещает блоки относительно друг друга, но может незначительно ошибаться при точном целевом расположении блоков. Основываясь на метрике strict F1, также очевидно, что предлагаемое решение лучше определяет направление, в котором необходимо построить фигуру из контекста диалога, по сравнению с другими решениями.

Очевидно, что предсказания с использованием ChatGPT приводят к хорошим результатам при использовании точных координат фигуры, но при этом дают низкое качество при использовании примитивов. Это иллюстрирует ограничения подхода с анализом нескольких примеров, при котором качество зависит от того, как формируется запрос. Если запрос на прогнозирование координат сформулировать относительно просто, передать все нюансы прогнозирования примитивов становится сложнее, и модель работает хуже в этом режиме. В то же время видно, что прогнозы, генерируемые с помощью дообученной модели Flan-T5, неизменно дают высокие результаты, независимо от формата данных. Хотя получены более высокие результаты для варианта L2S (ChatGPT), чем для варианта L2S (prim), предполагается, что добавление большего количества обучающих данных для последнего подхода может привести к повышению производительности. Однако в прогнозах, основанных на ChatGPT, есть ограничение по размеру запросу.

Метод прогнозирования на основе примитивов L2S (prim) неизменно превосходит исходный метод L2S (coords), который основан на базовой модели Flan T5, по всем показателям. Это преимущество в производительности объясняется тем, что фигуры, полученные с помощью предложенного подхода к объектной декомпозиции начальной задачи, содержат меньше ошибок при построении. Более того, можно заметить, что показатели метрик уменьшаются не так сильно при построении фигур, предсказанных по примитивам, в отличие от тех, которые основаны на предсказаниях координат. Эта тенденция подчеркивает большую надежность используемой объектно-центричной стратегии. Кроме того, поскольку созданная фигура разделена на параллелепипеды, а не на набор случайно разбросанных блоков, это упрощает задачу для агента, облегчая ее построение.

В данном разделе диссертационного исследования был представлен метод конструирования подзадач по языковой инструкции (L2S), являющийся новым подходом для эффективного следования инструкциями манипуляции различными объектами в специализированной симуляционной среде. Была продемонстрирована эффективность этого подхода в среде IGLU, разработанной на основе общеизвестной среды Minecraft, где целью агента является создание трехмерных фигур на основе текстовых описаний, предоставленных пользователем.

Метод L2S включает в себя два независимо обучаемых модуля наряду с промежуточным модулем менеджера задач. Задача языкового модуля заключается в преобразовании текстовых инструкций в определенную последовательность подзадач, в то время как модуль стратегии (получаемые на основе обучения с подкреплением) решает задачу навигации. Такая декомпозиция архитектуры в соответствии с подходом нейросимвольной интеграции NSLP позволяет структурировать решение задачи и способствует эффективному достижению агентом поставленных целей. Помимо того, он подходит для решения широкого спектра задач в различных условиях окружающей среды. Модульная структура также обеспечивает гибкость, необходимую для проведения экспериментов с языковой моделью без необходимости внесения изменений в архитектуру системы. Такая структура обеспечивает получение высококачественных прогнозов даже при ограниченном размере данных, в том числе благодаря дальнейшим усовершенствованиям с помощью ChatGPT и объектно-центрическому преобразованию данных в примитивные структуры.

Было продемонстрировано преимущество предложенного подхода по сравнению с решениями соревнования IGLU 2022. Необходимо отметить, что L2S обеспечивает превосходство по всем оцениваемым показателям, даже при меньшем количестве параметров модели.

Несмотря на то что проведенное исследование в основном сосредоточено на одной задаче из конкурса IGLU, предполагается, что архитектура L2S может послужить основой для решения других задач в визуально-языковых средах, где агенту необходимо выполнить несколько подзадач в рамках одного эпизода. Такие среды, как MineDojo [434] или Crafter [130], являются примерами таких сред. Они включают в себя различные подзадачи, такие как сбор ресурсов, размещение блоков и борьба с противниками в рамках одного

эпизода. В этом случае необходимо предсказать последовательность этих подзадач вместо предсказания размещения блоков/примитив в IGLU. В этих средах модуль стратегии также может быть независимо обучен в режиме целенаправленного обучения с подкреплением.

Необходимо отметить, что включение использования доступ к API проприетарной модели ChatGPT, используемого в целях аугментации набор данных IGLU, в дополнение к использованию других моделей и свободно лицензируемого кода в данной работе является некоторым ограничением данного подхода. Необходимо отметить, что отсутствие высокоэффективных инструктивных языковых моделей потенциально может служить ограничением продолжению исследований в этой области при воспроизведении предложенной техники аугментации данных.

4.5 Выводы

В данной главе диссертационного исследования были рассмотрены вопросы, связанные с реализацией нейросимвольного уровня архитектуры NSLP. Была проанализирована объектно-центричная парадигма, в которой в передаваемом когнитивному агенту наблюдении из среды выделяются отдельные распределенные группы признаков, называемых объектами. Выделяемые объекты описываются с помощью векторных представлений, именуемых слотами. Описаны две процедуры по эффективному обучению таких слотовых процедур: алгоритм SMM, использующий идею описания слотов в виде смеси гауссовых распределений, и алгоритм VQ-SA, предлагающий дополнительную квантизацию слотов и признаков, описывающих объекты в слотах. Оба подхода показывают преимущество в различных задачах (прогнозирование множества свойств и обнаружение объектов на изображении) анализа визуальных сцен.

Получаемые слотовые объектно-центричные представления могут быть использованы для построения уже графового представления сцены за счет включения в алгоритм кодирования наблюдения графовой нейронной сети. Получаемые представления сцен уже, в свою очередь, используются в архитектуре «актор-критик» в режиме обучения с подкреплением на основе модели. Предложенный в данной главе алгоритм ROCA, реализующий объектно-центричный вариант одновременного обучения и планирования в архитектуре NSLP, показывает отличные результаты при обучении стратегии в синтетических простых средах манипулирования объектами, превосходящие современные методы обучения с подкреплением на основе модели среды (такие как DreamerV3).

Наконец, в заключение данной главы было показано, что объектно-центричный подход оказывается эффективным и в конфигурации использования в архитектуре NSLP предобученной языковой модели для генерации высокоуровневого плана, состоящего из нескольких подзадач. Предложенный метод L2S, использующий также библиотеку навыков по навигации, полученную в результате предобучения в среде, показывает наилучшие результаты в задаче построения различных фигур по языковым инструкциям в постановке

соревнования IGLU 2022 по сравнению с лучшими текущими решениями этой задачи (алгоритмами BrainAgent и Pegasus).

Глава 5. Обучение и планирование в системах управления мобильными робототехническими платформами

В данной главе представлены робототехническая реализация архитектуры STRL+NSLP и прикладные примеры решения задач навигации мобильной робототехнической платформы и маневрирования беспилотного автомобиля. Основные результаты, изложенные в данной главе, были опубликованы в [9; 49; 73; 75] (раздел 5.2), [3; 76] (раздел 5.1), [27; 29; 30] (раздел 5.3).

5.1 Программная реализация архитектуры NSLP для мобильного робота

Данный раздел диссертационного исследования посвящен рассмотрению интеграции конкретных реализаций различных блоков и модулей архитектуры NSLP в рамках робототехнической операционной системы (Robot Operating System, ROS). Данная программная реализация была названа STRL-Robotics. Рассмотрены подробности связывания сенсорной информации с выходами более низкого уровня управления в архитектуре STRL при решении задачи мобильной манипуляции на реальном роботе Husky. Основные результаты, изложенные в данном разделе, представлены в публикациях [76].

Мобильные манипуляторы – робототехнические системы, состоящие из платформы и закрепленного на ней манипулятора – могут применяться для решения различных задач в человеко-ориентированных средах в условиях коллaborации с людьми. Частным примером прикладных коллаборативных робототехнических систем являются роботы-ассистенты, используемые для поддержки людей с ограниченными возможностями и позволяющие им выполнять механические действия, привычные для здоровых людей, но недоступные при тяжелых заболеваниях нервной системы или опорно-двигательного аппарата [110; 366; 493]. С более общей точки зрения, коллаборативные мобильные манипуляторы могут применяться для автоматизации решения задач в человеко-ориентированных средах в качестве курьеров, ассистентов, грузчиков и т.п. В настоящее время такие системы активно развиваются [105; 366; 437]. В качестве примеров существующих исследовательских и коммерческих систем можно указать Everyday Robot от Google, линейку роботов TIAGo от фирмы PAL Robotics, мобильный манипулятор Fetch от производителя Fetch Robotics, Handle от фирмы Boston Dynamics, Stretch от компании Hello Robot. Кроме того, мобильный манипулятор можно собрать из доступных мобильной платформы и манипулятора: например, в рамках данной работы применялась система, составленная из платформы Husky UGV от фирмы Clearpath, манипулятора UR5 от компании Universal Robots и набора сенсоров, закрепленных на платформе и манипуляторе (рисунок 5.1).

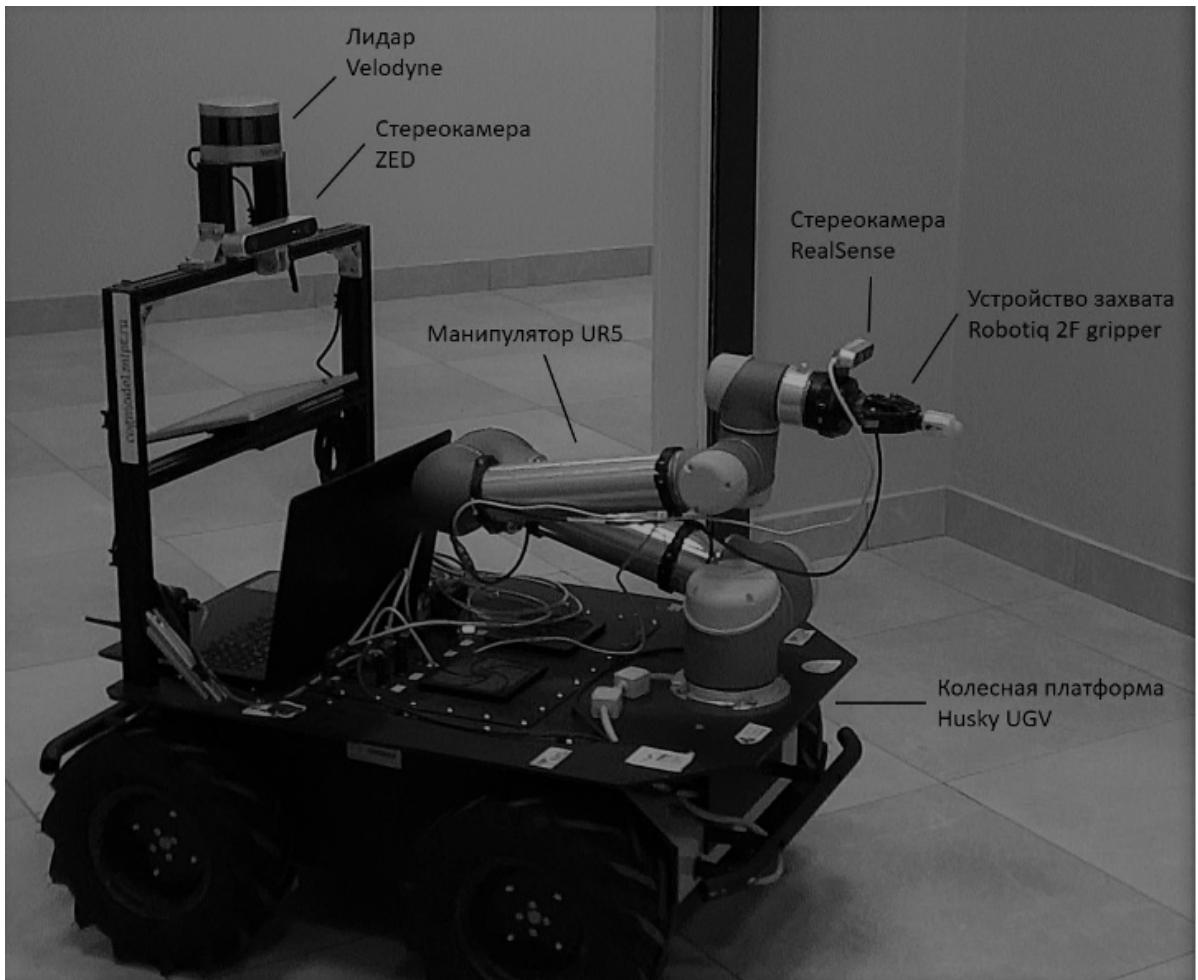


Рисунок 5.1 — Используемая в экспериментах робототехническая система.

Мобильные манипуляторы, функционирующие в человеко-ориентированных средах, должны решать множество разнотипных задач, которые требуют выполнения комплекса действий в автономном режиме с бесшовным переходом от одной подзадаче к другой. Это обуславливает актуальность разработки архитектуры для решения комплексных задач колаборативным мобильным манипулятором. Отдельные подзадачи, решаемые в процессе выполнения действий (восприятие среды, навигация, распознавание объектов, планирование движения, управление движением, манипулирование объектами), могут быть хорошо проработаны в научной литературе; для их решения существуют специализированные алгоритмы и программные модули. Отдельной подзадачей становится объединение модулей в единую архитектуру с выбором и синтезом тех методов, подходов и алгоритмов, которые показывают наибольшую эффективность именно при решении комплексной задачи. Кроме того, при объединении алгоритмов возникает проблема разделения вычислительных ресурсов между ними, поскольку многие алгоритмы для решения робототехнических задач (особенно связанные с обработкой данных визуальных сенсоров) являются ресурсоемкими, в то время как встроенное аппаратное обеспечение робототехнических систем зачастую ограничено в вычислительном плане.

В данном разделе диссертационного исследования предлагается общее решение комплексной задачи обеспечения мобильности колаборативной робототехнической системы

в многоэтажном офисном здании, оснащенном лифтами. Представлена реализация архитектуры NSLP в виде иерархической интеллектуальной модели системы управления для решения комплексных робототехнических задач, STRL-Robotics, основанной на общей архитектуре управления поведением когнитивных агентов STRL [61; 89].

5.1.1 Основные компоненты платформы STRL-Robotics

Как уже описывалось ранее в разделе 2.2, название архитектуры STRL [88] расшифровывается как «Strategic, Tactical, Reactive Layered» и отображает тот факт, что модель состоит из трех основных уровней: стратегического, тактического и реактивного. Первый уровень отвечает за реализацию высокоуровневых интеллектуальных функций: на нем определяется последовательность действий, которые должен выполнить робот для решения некоторой задачи [44; 60]. Второй — за построение сенсорной ситуации и навигацию объекта (локализация на карте и планирование траектории) в среде в процессе выполнения задачи. На реактивном уровне контролируется выполнение отдельных операций, необходимых для следования объекта по спланированной траектории с учетом динамических ограничений и обратных связей.

Архитектура STRL была изначально предложена для координации действий группы интеллектуальных агентов, например, беспилотных летательных аппаратов (БПЛА) [13]. Специфика мобильных манипуляторов по сравнению с БПЛА состоит в том, что они представляют собой композицию двух механических систем (платформы и манипулятора), которые могут управляться независимо. В рамках данного раздела предполагается, что мобильная платформа применяется для перемещения робототехнической системы в человеко-ориентированной среде, а манипулятор — для взаимодействия с объектами человеко-ориентированной среды. Впоследствии такая схема может быть расширена на задачи, в которых при взаимодействии с объектами среды используется не только манипулятор, но и мобильная платформа.

Общая схема взаимодействия компонентов системы управления при обеспечении мобильности робота в человеко-ориентированной среде приведена на рисунке 5.2. Управление движением как платформы, так и манипулятора должно учитывать данные от сенсоров системы, которые преобразуются алгоритмами компьютерного зрения в представление большего уровня абстракции на тактическом уровне модели STRL-Robotics (построение сенсорной ситуации). Если в [88] это преобразование использовалось только для задач навигации, то настоящей работе добавляется также задача распознавания и позиционирования объектов среды, с которыми взаимодействует коллaborативная робототехническая система. Определение действий, выполняемых робототехнической системой, осуществляется отдельно на тактическом (определение геометрического пути, который должна пройти система для достижения поставленной цели) и на реактивном (определение управляющих команд, обеспечивающих движение системы по геометрическому

пути) уровнях модели. Таким образом, можно выделить шесть основных подзадач, решаемых модулями системы:

1. построение карты среды и локализация на ней,
2. планирование геометрического пути платформы между заданными точками на карте,
3. управление движением мобильной платформы по геометрическому пути,
4. распознавание и локализация объектов, с которыми взаимодействует манипулятор,
5. планирование движения мобильного манипулятора для решения задачи, подразумевающей взаимодействие с объектами среды,
6. управление движением мобильного манипулятора для решения задачи, подразумевающей взаимодействие с объектами среды.

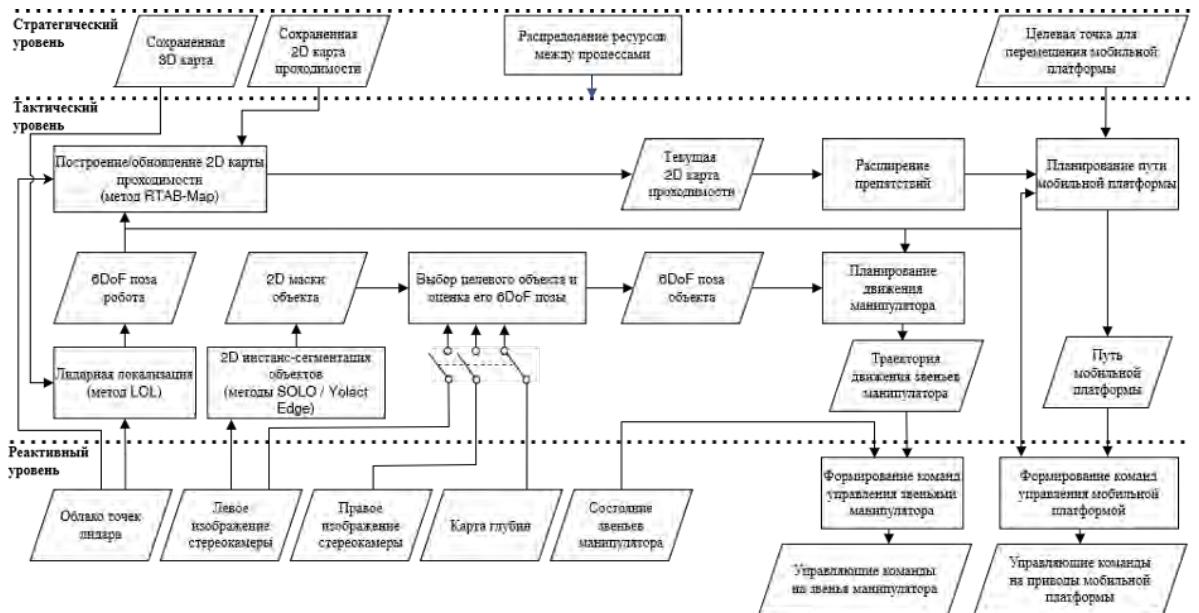


Рисунок 5.2 — Взаимодействие программных модулей системы в рамках предлагаемой архитектуры STRL-Robotics.

Первые три подзадачи используются для обеспечения мобильности коллаборативной робототехнической системы. Их решение последовательно описано в следующих трех параграфах этого раздела. Эти подзадачи можно назвать «универсальными» в том смысле, что их решение будет одинаковым для различных задач, подразумевающих перемещение системы между различными отстоящими далеко друг от друга точками среды. Этой универсальностью не обладают следующие три подзадачи: алгоритмы, применяемые для их решения, будут зависеть от того, как и с какими объектами среды взаимодействует робототехническая система. В заключение этого раздела подробно рассмотрен частный пример такого взаимодействия: использование лифта. Распознаваемыми объектами в этом случае будут кнопки лифта, а движения манипулятора планируются таким образом, чтобы нажимать нужные кнопки.



Рисунок 5.3 — Структурная схема процесса реконструкции трехмерной карты местности.

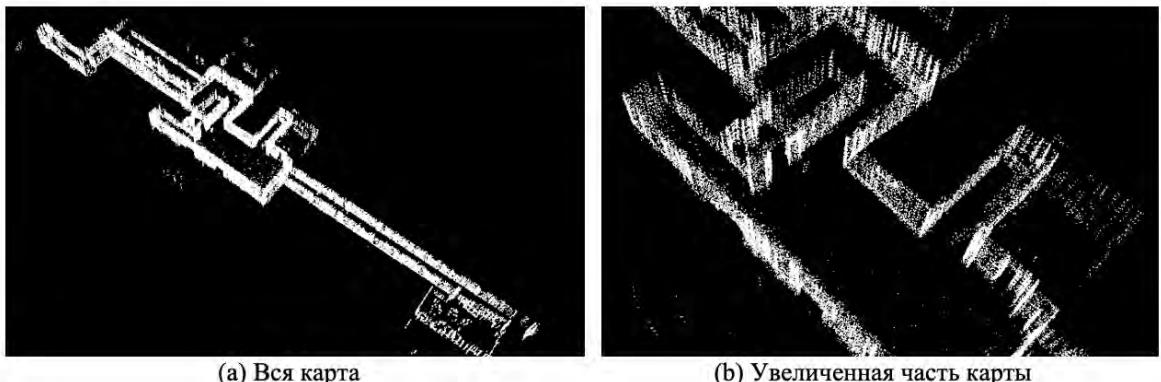


Рисунок 5.4 — Пример построенной лидарной карты этажа офисного здания (пятый этаж корпуса Физтех.Цифра МФТИ). Для лучшей визуализации показаны только угловые точки.

Слева — вся карта, справа — увеличенная часть карты.

5.1.2 Реализация сенсорных блоков

Надежное позиционирование робота в пространстве (определение его трехмерных координат и углов ориентации) достигается за счет осуществления локализации на предварительно построенной трехмерной карте местности с помощью облаков точек многолучевого лидара [148]. Для построения карты необходимо совершить одиночный проезд по территории в ручном режиме управления и собрать набор последовательных облаков точек. Эти данные используются в методе одновременной локализации и картирования (Simultaneous Localization And Mapping, SLAM) для реконструкции трехмерной карты в виде единого облака точек (см. рисунок 5.3). Последующая локализация на карте осуществляется посредством сопоставления нескольких последних наблюдаемых облаков точек и облака точек карты (см. рисунок 5.4).

Реконструкция трехмерной карты местности осуществляется с помощью лидарного метода одновременной локализации и картирования A-LeGO-LOAM [561], имеющего реализацию с открытым исходным кодом¹. Схема подхода изображена на рисунке 5.3. Работа метода включает в себя два этапа: вычисление одометрии робота (трехмерных координат, углов ориентации и скорости их изменения) и построение карты. Вычисление одометрии осуществляется путем сопоставления текущего и предыдущего лидарных облаков на основе извлеченных из них угловых и планарных точек. Данный этап обладает низкой вычислительной сложностью, однако на больших отрезках времени ошибка накапливается.

¹<https://github.com/jyakaranda/A-LeGO-LOAM>

Для увеличения точности выполняется построение карты в режиме реального времени путем накопления лидарных облаков точек. При добавлении облака точек к карте осуществляется его сопоставление с уже построенной картой на основе угловых и планарных точек текущего облака и карты. Пример результата построения трехмерной карты (размером 114×17 метров) приведен на рисунке 5.4; плоскости стен в помещениях были реконструированы достаточно точно.

Локализация на реконструированной карте осуществляется с помощью метода пакетной обработки лидарных сканов (scan batch processing), предложенного в [658], и имеющего реализацию с открытым исходным кодом в виде алгоритма LOL (Lidar Odometry and Localization)². Из лидарного облака аналогично методу A-LeGO-LOAM извлекаются признаки (угловые и планарные точки). Угловые точки сопоставляются с линиями границ на сохраненной карте, планарные — с плоскими участками. Локализация осуществляется за счет минимизации расстояний между сопоставленными точками облака точек и сохраненной карты. Модуль локализации, в целом, аналогичен модулю лидарной одометрии, используемому на этапе реконструкции карты. Отличие заключается в том, что локализация осуществляется на основе не одного скана, а набора последовательных сканов, которые объединяются в одно облако точек с использованием данных лидарной одометрии. За небольшой промежуток времени (несколько секунд) ошибка в определении изменения положения с помощью лидарной одометрии мала, что и позволяет достаточно точно объединить несколько сканов.

Лидарные облака точек и результат локализации на статической, предварительно построенной трехмерной карте местности используются сначала для построения начальной двумерной карты проходимости с помощью метода RTAB-Мар [369] с открытым исходным кодом³, а затем и для ее обновления в процессе автономного движения робота. Наличие предварительно построенной карты проходимости позволяет осуществлять глобальное планирование, а ее обновление — корректно обрабатывать ситуации изменения среды, такие как появление новых статических или движущихся объектов. Чтобы не использовать избыточные возможности подхода RTAB-Мар, который реализует полноценный визуальный SLAM с учетом лидарных данных, из него был выделен модуль, который используется для построения карт препятствий⁴. Этот модуль был доработан в виде ROS-узла с возможностью сохранения, загрузки и обновления карты препятствий. Следует отметить, что разработанный модуль локализации обеспечивает среднюю абсолютную ошибку позиционирования менее 15 сантиметров и ошибку определения ориентации менее одного градуса. Благодаря использованию сохраненной заранее карты модуль лишен проблем накопления ошибки со временем, что позволяет строить достаточно точную карту препятствий.

²<https://github.com/RozDavid/LOL>

³<https://github.com/introlab/rtabmap>

⁴Узел occupancy_grid_builder https://github.com/andrey1908/rtabmap_ros

5.1.3 Реализация блоков планирования

Для решения задачи планирования движения существует множество различных подходов: в частности, их подразделяют на клеточные подходы [360; 535] и на основе выборки [367; 494]. В различных постановках предлагаются разные требования к алгоритмам. В рассматриваемом случае требования выдвигаются следующие: высокая скорость работы, возможность перепланирования, интегрируемость и управляемость, возможность быстрой реконфигурации. Поскольку в данной задаче не учитывались кинематические ограничения, для управления движением агента был выбран алгоритм планирования Theta* [617], позволяющий реализовывать движение под любым углом. Theta* является модификацией широко известного алгоритма A*. Для решения задачи планирования A* фокусирует поиск пути, используя эвристическую функцию, которая оценивает расстояние от исследуемого состояния до целевого. В различных средах эвристическая функция имеет различный вид, например, в среде, где допустимы только ортогональные перемещения, эвристическая функция будет равна расстоянию Манхэттена, а в так называемой 8-связной сетке — диагональному расстоянию. Theta* наследует этот принцип поиска, но его основное отличие заключается в возможности планировать траекторию не только в виде переходов между соседними клетками, но и в виде больших «скакков» между любыми двумя узлами карты, между которыми есть прямая видимость. Это приводит к тому, что эвристическая функция для Theta* должна иметь вид декартового расстояния. В остальном алгоритм планирования Theta* повторяет алгоритм A*.

Поскольку в процессе движения среда может измениться (появляются ранее неизвестные препятствия или некоторое динамическое препятствие), необходимо обеспечить процесс перепланирования. В рассматриваемом случае благодаря высокой скорости алгоритма агент строит путь заново из своего текущего положения в ту же целевую точку с некоторой регулируемой частотой. Также при планировании необходимо учесть габариты робота, поскольку кратчайший маршрут может проходить близко к препятствиям, а линия маршрута соответствует перемещению геометрического центра робота. Для решения этой подзадачи также существуют различные подходы, среди которых был выбран следующий: размеры препятствий на карте увеличиваются путем раздвижения их границ на величину, равную внешнему радиусу робота R1 (рисунок 5.5а). Это гарантирует, что траектория, построенная непосредственно около препятствия, будет безопасна, поскольку при любой ориентации обеспечено отсутствие столкновения.

В ходе экспериментов было замечено некорректное поведение, при котором робот, находящийся около препятствия, не может начать планировать траекторию, поскольку сенсоры определяют его положение внутри этого увеличенного препятствия, даже если он находится на достаточном расстоянии. Это связано с определенной приборной погрешностью лидара, ошибками округления в коде и т.д. В связи с этим был введен еще один уровень увеличения препятствий, соответствующий внутреннему радиусу робота R2. Алгоритм был настроен таким образом, что новая траектория строится в обход увеличенного размера R1,

однако, если в некоторый момент агент находится в области между R2 и R1, он находит кратчайший выход в свободную зону (рисунок 5.5b), и дальнейшая траектория проходит по свободной зоне. Это нововведение также было важно для процесса перепланирования, так как во многих траекториях, содержащих огибание препятствий, возникала описанная выше ситуация. Таким образом, агент либо сразу же выезжает в безопасную зону, где разрешены любые маневры, либо, в случае некорректного начального положения, так и не начинает движение. Для реализации описанного поведения был реализован ROS-узел, в который был включен алгоритм Theta*.

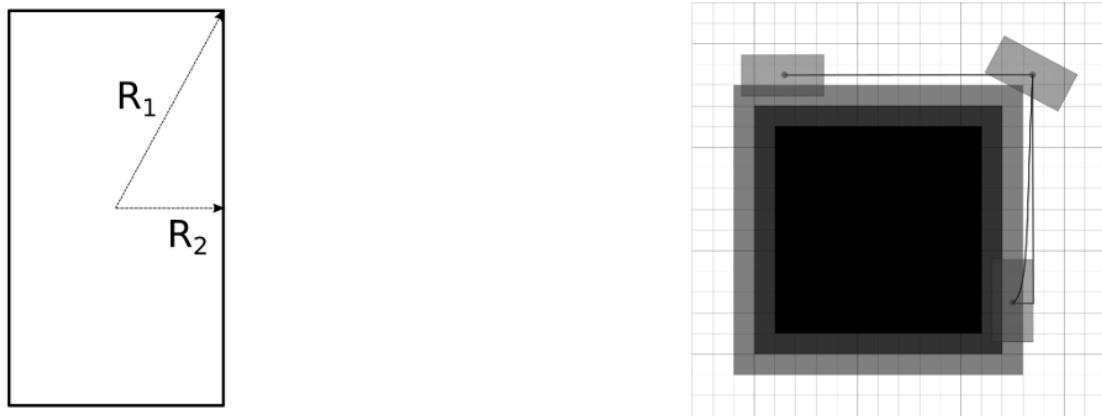


Рисунок 5.5 — Демонстрация планирования с увеличенными препятствиями. На рисунке справа разные уровни увеличения препятствия показаны разными оттенками серого. Слева — геометрические размеры агента, используемые для увеличения препятствий, справа — спланированная траектория при условия нахождения робота в запрещенной области. Красным цветом отмечен геометрический путь, синим — примерное начало траектории движения робота.

Для решения задачи управления движением платформы был предложен новый алгоритм. Особенностью алгоритма является уменьшение среднего и максимального отклонений от заданного пути по сравнению с рассмотренными аналогами [260; 626] при сопоставимых временных затратах на реализацию передвижения. Расстояние отклонения определяется как длина высоты, опущенной от положения робота до ближайшего сегмента пути. Алгоритм и его аналоги были протестированы на мобильной робототехнической платформе Husky (см. рисунок 5.1). Робот может принимать желаемые линейную и угловую скорости в качестве управляющих команд, реализацию которых осуществляет встроенный регулятор. В ходе экспериментов максимальные по абсолютной величине значения для этих сигналов определены как 1 м/с и 2 рад/с соответственно.

Целью алгоритма следования пути является реализация передвижения робота по заданному пути [626], т.е. в рассматриваемом случае — по отрезкам прямых, найденных на этапе планирования. Необходимо подчеркнуть, что путь не определяет скорость перемещения робота. Также важно отметить, что в рассматриваемом алгоритме ориентация робота всегда направлена на конечную точку B текущего сегмента (см. рисунок 5.6а). Для минимизации отклонения робота от желаемого пути d был использован модифицированный алгоритм,

в котором в каждый момент движения допускается добавлять искусственную точку E на отрезке пути так, чтобы ориентация робота была направлена на точку E вместо B (см. 5.6b). Законы управления скоростями задаются системой [626]:

$$v = \sqrt{(x_B - x)^2 + (y_B - y)^2}, \omega = \operatorname{arctg}(\varphi_E - \varphi), \quad (5.1)$$

где x, y, φ — координаты центра масс и ориентация робота в глобальной системе координат; x_B, y_B — положение точки B ; φ_E — ориентация точки E в глобальной системе координат.

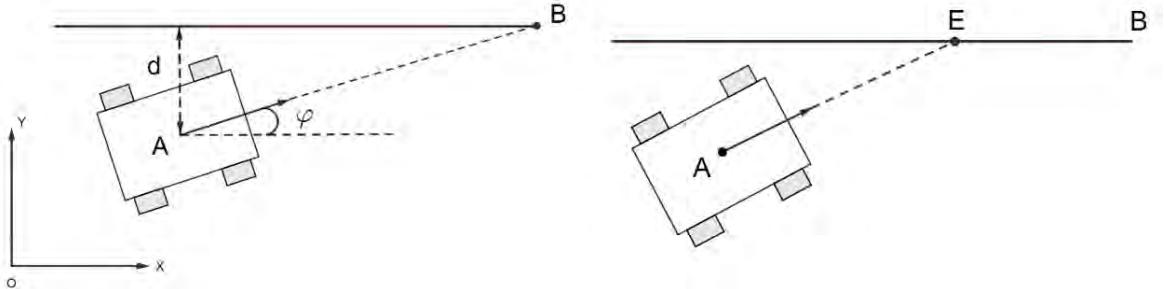


Рисунок 5.6 — Работа алгоритма следования пути. Слева — исходная версия, справа — модифицированная версия.

Целью базового алгоритма следования является реализация заданной траектории движения. Отличие траектории от пути в том, что она определяет скорость перемещения робота, т.е. его желаемое состояние определяется как функция времени. Поскольку найденный на этапе планирования путь не содержит информации о скорости, она определяется на реактивном уровне архитектуры. Скорость по траектории выбрана постоянной, и ее значение меньше максимальной скорости робота. Алгоритм следящего управления основывается на работе [260] и базируется на методе Ляпунова. Законы управления линейной и угловой скоростями задаются следующим образом:

$$v = v_d \cos e_3 + k_1 e_1, \omega = \omega_d + v_d k_2 e_2 + k_3 \sin e_3, \quad (5.2)$$

где v_d, ω_d — линейная и угловая скорости траектории; e_1, e_2, e_3 — невязка между фактическим и желаемым положением робота в каждый момент времени в локальной системе координат; k_1, k_2 и k_3 — некоторые положительные коэффициенты.

Закон управления предложенного метода определяется формулами 5.2. Модификация заключается в формировании траектории, которую отслеживает робот. Если по какой-либо причине робот отклонился от подвижной точки, определяющей, согласно заданной траектории, желаемое положение робота в каждый момент времени, то алгоритм слежения по работе [260] будет стараться минимизировать расстояние до этой точки. Последнее может привести к сильному отклонению от построенного планировщиком пути, даже если траектория из исходной точки следует точно по этому пути, что, в свою очередь, может привести к столкновению с препятствиями (см. рис. 5.7а).

Для минимизации отклонения робота от пути предложен подход, основная идея которого схожа с идеей модификации алгоритма следования по пути, а именно, она заключается в создании новой траектории, которая следует вдоль отрезка прямой,

заключенного между положением робота и искусственно добавленной точки E , находящейся на исходном пути (см. 5.7b). Робот начинает двигаться по этой траектории. Когда он достигает точки E_1 , операция повторяется: строится новая траектория, проходящая через отрезок прямой между текущим положением робота и новой точкой E_2 и так далее.

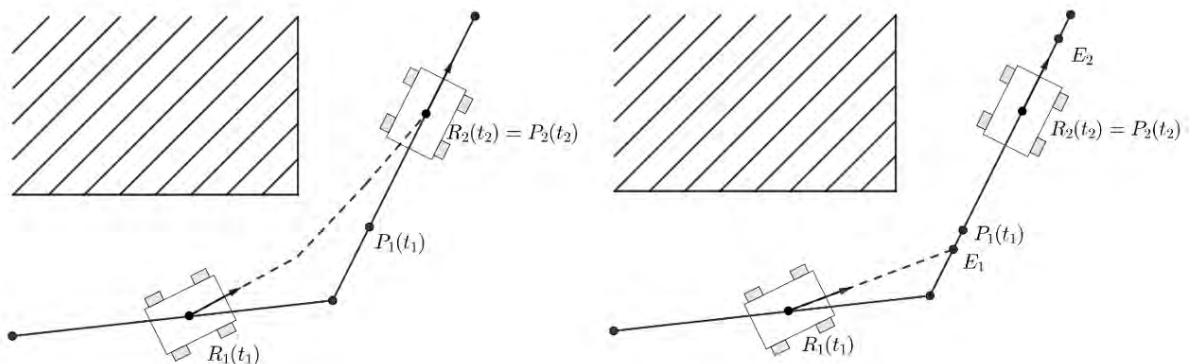


Рисунок 5.7 — Работа алгоритмов следования по траектории. Слева — алгоритм из работы [260], справа — модифицированная версия. Здесь R_1, R_2 и P_1, P_2 — фактическое и желаемое положения робота в моменты t_1 и t_2 соответственно, $t_1 < t_2$.

5.1.4 Модельный пример с использованием лифта

Использование лифта робототехническими системами рассматривалось в ряде работ [108; 126; 193; 223; 274; 504; 644]. Для вызова лифта и выбора этажа может устанавливаться беспроводная связь между контроллерами робота и лифта [274], однако такой подход требует развертывания специальной системы связи на базе системы управления лифтом, что не требуется, если робот может нажимать кнопки на панелях лифта, как это обычно делает человек. Если мобильному манипулятору ставится задача навигации между точками в здании, расположенными на разных этажах, то для ее решения нужно выполнить следующие действия: проехать от начальной точки до лифтового холла, вызвать лифт, заехать в лифт, нажать кнопку нужного этажа, выехать из лифта на нужном этаже, проехать от лифтового холла до конечной точки. Задачи навигации колесной платформы при этом могут решаться методами, описанными в предыдущих разделах. Вызов лифта и выбор нужного этажа в лифте решается как задача взаимодействия мобильного манипулятора с объектом, включающая подзадачи распознавания этих объектов (кнопок на панели лифта) и управления манипулятором в процессе взаимодействия с объектом (нажатия на кнопку). Подходы к решению этих подзадач описаны ниже.

Распознавание и позиционирование объектов — кнопок лифта

Современные решения по распознаванию объектов на изображениях основываются на глубоких нейронных сетях. Исследователи могут выбирать из широкого спектра архитектур в зависимости от аппаратных ограничений. Авторы работы [410] использовали сеть обнаружения SSD [590] для распознавания классов и FCN, чтобы получить маски кнопок лифта, однако они не исследовали задачу определения положения. Исследователи в [329] использовали архитектуру YOLO [525], чтобы выполнять распознавание по монокулярной камере и 2D-лидар для определения позы. Авторы работы [107] собрали крупномасштабный набор данных и предложили модель OCR-RCNN. Основываясь на обширных сравнительных экспериментах, исследователи выяснили, что нейронные модели легко переобучаются из-за проблемы диспропорции классов. В работе [126] авторы улучшили OCR-RCNN и представили роботизированную систему для оценки положения кнопки с помощью RGB-D камеры с учетом дисперсии (неопределенности) измерений. Дополнительно в работе используется детектор линий (LSD), который, как выяснилось, может снижать надежность системы в сложных сценариях изменения освещения.

Большой объем размеченных данных является ключевым фактором для построения надежных моделей для распознавания объектов. В случае с кнопками лифта имеется крупномасштабный набор данных, который содержит 3718 изображений с различными классами кнопок и панелей, позами и масками сегментации. В общем доступе, однако, находятся только изображения с ограничивающими прямоугольниками. Кроме данного набора, известны несколько непубличных наборов данных [329; 410].

Для распознавания кнопок лифта было решено остановиться на задаче сегментации объектов (инстанс-сегментации), так как маски по сравнению с ограничивающими прямоугольниками позволяют точнее определять границы объектов и тем самым точнее разделять отдельные экземпляры. Из-за фактического отсутствия общедоступного набора данных с аннотациями сегментации для экспериментов изображения были размечены вручную. Эти данные публично доступны⁵. Разработанный набор данных ElevatorButtons содержит 150 размеченных изображений из более чем 70 различных лифтов, сохраненных в формате, пригодном для работы в инструменте разметки OpenCV CVAT⁶. Он разделен на тренировочную, валидационную и тестовую выборки в пропорции 8:1:1 в формате MS COCO [432].

Для оценки 3D-позиции кнопки на мобильной платформе можно использовать RGB-D камеру Zed2 или 16-лучевой лидар Velodyne VLP-16. В результате проведенного анализа был выбран первый способ, так как лидар имеет чрезмерно разреженное облако для работы с небольшими объектами. Для оценки позы кнопки было принято несколько допущений: стены прямые и перпендикулярны полу, область вокруг кнопок имеет тот же вектор нормали, что и сами кнопки. В ходе экспериментов с имеющейся мобильной платформой

⁵РМРеРçР«СГРуРеР,,Р,,СКСЖElevatorButtons.https://github.com/linukc/elevator_buttons

⁶<https://cvat.org/>

Husky A200 было замечено, что данные о глубине для кнопок и областей рядом часто отсутствуют. Это вызвано тем, что материал, из которых они изготовлены, и области вокруг однотонные и порождает мало текстур на изображении — для соответствующих пикселей аналитические методы реконструкции карт глубин не могут точно определить значения глубины. В результате формируемая встроенными алгоритмами камеры Zed2 карта глубин достаточно зашумлена. В качестве альтернативы предложено исследовать возможности получения 3D-позы объекта-кнопки (с шестью степенями свободы) с помощью триангуляции координат распознанного объекта на левой и правой камере.

После получения координат в системе изображения (u_x, u_y) 3D-поза центра кнопки (X_c, Y_c, Z_c) рассчитывается с помощью глубины z и известных внутренних параметров камеры (фокусных расстояний f_x, f_y и координат оптического центра c_x, c_y) следующим образом:

$$X_c = \frac{1}{f_x} (u_x - c_x) z, \quad Y_c = \frac{1}{f_y} (u_y - c_y) z, \quad Z_c = z. \quad (5.3)$$

Для устранения выбросов значений глубин z , возникающих из-за ошибок встроенного алгоритма реконструкции карт глубин камеры Zed2 или ошибок триангуляции на основе неточных результатов обнаружения областей объектов применяется фильтрация 20 последних измерений на основе гистограммы глубины с шагом 0,005 м. В результате выбирается среднее значение глубины в интервале, в который попало наибольшее число измерений.

Вектор нормали (ориентация) объекта определяется как векторное произведение двух векторов, лежащих в плоскости кнопки по формуле:

$$\mathbf{n} = \mathbf{CA} \times \mathbf{CB} = (X_a - X_c, Y_a - Y_c, Z_a - Z_c) \times (X_b - X_c, Y_b - Y_c, Z_b - Z_c), \quad (5.4)$$

где для составления векторов используются пространственные координаты центра $C(X_c, Y_c, Z_c)$ и двух точек на ограничивающем прямоугольнике кнопки: точки $A(X_a, Y_a, Z_a)$ — середины между верхней левой и верхней правой вершинами и точки $B(X_b, Y_b, Z_b)$ — середины между верхней правой и нижней правой вершинами, которые рассчитываются аналогично 3D-координатам центра по формуле 5.3.

Нажатие на кнопку робототехническим манипулятором

Подходы к управлению при нажатии на кнопку зависят от конструкции манипулятора: количества звеньев и типов связей между ними (связи могут обеспечивать вращательное либо поступательное движение звеньев). В ряде работ [193; 223; 644] рассматриваются декартовые манипуляторы, состоящие из трех последовательно движущихся звеньев, каждое из которых обеспечивает одну степень свободы в декартовых координатах. Подходы к управлению такими роботами специфичны и не могут быть применены на колаборативных манипуляторах с вращательными связями. В работе [108] использован плоский манипулятор

с тремя степенями свободы и определено решение инверсной кинематики для вычисления обобщенных координат при нажатии на кнопку. В работе [126] применялась камера, закрепленная на рабочем органе манипулятора. Команды манипулятора определялись таким образом, чтобы привести изображение с камеры к нужному виду. На каждом цикле управления рассчитывалось текущее направление на кнопку со стороны камеры, определялось желаемое положение кнопки на изображении для следующего цикла и выполнялось движение манипулятора, обеспечивающее это положение. В работе [504] использовался антропоморфный робот с трехзвенной рукой. Движение манипулятора разделялось на два этапа: перемещение в окрестность кнопки и непосредственно нажатие. Оба этапа выполнялись как прямой переход между состояниями в обобщенных координатах.

В целом, планирование движения манипулятора между заданными точками может осуществляться различными методами на основе выборки, например, RRT-Connect [367]. Однако в вышерассмотренных работах [108; 126; 504] они не применялись — вместо этого использовались прямые переходы между состояниями робота. Методы планирования на основе выборки достаточно ресурсоемки, и основное преимущество их применения — избегание коллизий со средой. Перед панелью лифта обычно есть свободное от препятствий пространство и можно перемещать манипулятор между заданными состояниями по прямой, не опасаясь коллизий.

Современные коллаборативные манипуляторы (в частности, используемый в экспериментах UR5) зачастую имеют опцию управления движением не только в обобщенных, но и в декартовых координатах: можно задать манипулятору некоторые декартовы координаты, и рабочий орган переместится в эту точку по прямой. Задача инверсной кинематики при этом решается встроенными контроллерами манипулятора. Пространство состояний, в котором возможно такое управление, меньше, чем для обобщенных координат, поскольку существует область сингулярности манипулятора, в которой встроенные контроллеры не могут эффективно решать задачу инверсной кинематики. При движении платформы манипулятор обычно находится в сложенном состоянии, которое является сингулярным (например, на рисунке 5.1). В связи с этим было решено при планировании движения колесной платформы к лифтовой панели задавать конечное положение таким образом, чтобы кнопки лифта не находились в зоне сингулярности манипулятора.

Исходя из вышесказанного была предложена следующая последовательность действий, обеспечивающая нажатие на кнопки лифта с помощью последовательности прямых переходов.

1. Платформа занимает нужную позицию относительно панели лифта. Эту позицию можно задать заранее исходя из доступной информации о расположении кнопочных панелей в лифтовых холлах и лифтах здания. Данная точка является конечной точкой при планирование пути на тактическом уровне.
2. Манипулятор перемещается в некоторое положение в обобщенных координатах, удовлетворяющее следующим условиям. Во-первых, лифтовая панель в этом положении находится в поле зрения манипулятора. Во-вторых, для этого положения кнопки лифта будут достижимы при задании движения к ним в декартовых

координатах. В-третьих, движение в заданную позицию не приведет к столкновению с окружающими объектами. Это движение соответствует первому этапу из работы. Такое положение можно задать заранее исходя из доступной информации о расположении панелей в лифтах здания и соответствующих им позициях колесной платформы.

3. Производится распознавание кнопок лифта, выбирается нужная кнопка, и определяется декартова позиция рабочего органа, в которой он находится прямо перед кнопкой.
4. Рабочий орган перемещается в позицию, определенную на предыдущем шаге.
5. Рабочий орган движется к кнопке, пока датчик усилия не известит о нажатии. Датчик усилия выдает трехмерный вектор силы, действующей на рабочий орган. Составляющая этого вектора, коллинеарная направлению движения рабочего органа, показывает, с какой силой среда сопротивляется движению робота. Можно экспериментально определить пороговые значения, свидетельствующие о нажатии кнопки. Например, в ходе экспериментов, описанных в следующем разделе, пороговое значение было определено равным 50 ньютонам. Для менее тугих кнопок значение будет меньше.
6. Манипулятор возвращается в позицию, определенную на шаге 3, а затем складывается в исходное положение.

Такой сценарий нажатия позволяет возложить управление движением на встроенные контроллеры манипулятора за исключением распознавания факта нажатия на кнопку. По этой причине в данном разделе планирование пути манипулятора и управление движением по нему рассмотрены как единая подзадача. При более сложных постановках задач манипуляции можно выделить отдельный модуль управления движением манипулятора.

Общая ROS-реализация архитектуры управления

Разработанные модули и предложенная архитектура были реализованы на экспериментальной платформе, приведенной на рисунке 5.1. Взаимодействие компонентов осуществлялось через среду Robotic Operating System (ROS). Для оценки предложенных методов были проведены эксперименты по использованию лифта робототехнической системой. Перед этим проводились предварительные эксперименты, направленные на сравнение подходов к управлению колесной платформой и распознаванию кнопок лифта.

Алгоритмы управления движением и следования по траектории применялись к мобильному роботу Husky для десяти одинаковых путей (см. рисунок 5.8). Ниже представлены результаты для модифицированных алгоритмов следования по пути (см. табл. 16) и слежения (см. табл. 17). Исходный алгоритм слежения из работы [260] приводил к существенным отклонениям от построенных планировщиком путей, что в некоторых случаях сопровождалось столкновениями со стенами. Поэтому результаты экспериментов

для данного алгоритма не приводятся. В таблицах v_{max} — максимально допустимая скорость движения робота, d — среднее отклонение робота от пути, d_{max} — среднее максимальное отклонение робота от пути, t — среднее время прохождения. Средние значения вычислялись по результатам проведения четырех экспериментов, в которых последовательно проезжались все десяти путей. В таблице 17 параметр v_d — это желаемая постоянная скорость движения робота.

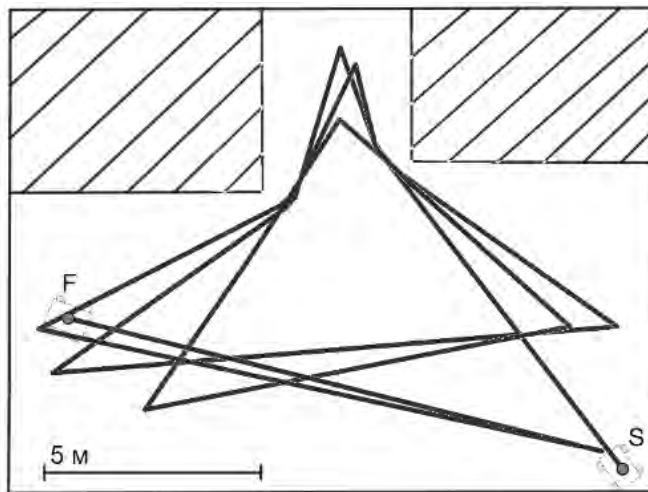


Рисунок 5.8 — Десять последовательных путей, использованных для экспериментов (разные цвета использованы для лучшей различимости. S и F — точки старта и финиша).

Таблица 16 — Результаты для алгоритма следования по пути.

v_{max} (м/с)	d (м)	d_{max} (м)	t (с)
0.65	0.0344	0.1127	256.8
0.75	0.0376	0.1187	242.8
0.85	0.0385	0.1186	228.6
0.95	0.0444	0.1296	220.6
1	0.0561	0.1527	219.2

Таблица 17 — Результаты для предложенного алгоритма слежения.

v_{max} (м/с)	v_d (м/с)	d (м)	d_{max} (м)	t (с)
0.65	0.60	0.024	0.1099	271.55
0.75	0.65	0.023	0.1062	263.9
0.85	0.75	0.022	0.1061	253.4
0.95	0.85	0.0223	0.1016	243
1	0.95	0.0222	0.1132	235.8

Для распознавания кнопок лифта были выбраны два метода сегментации объектов: SOLOv2 [582] и YolactEdge [653]. Эксперименты проводились с разработанным набором данных ElevatorButtons. Нейронная сеть SOLOv2 обучалась с использованием базовой модели ResNet-50 в течение 140 эпох с оптимизатором SGD и линейно изменяющимся

коэффициентом скорости обучения с начальным значением 0.001. Коэффициент уменьшался в десять раз на 100 и 130 эпохах. Метод YolactEdge обучался в течение 100 эпох также с аналогичной базовой моделью ResNet-50, с оптимизатором SGD и начальным коэффициентом скорости обучения равным 0.0005. Коэффициент уменьшался в десять раз на 70 и 90 эпохах.

В табл. 18 приведены количественные результаты обучения моделей. Эксперименты проводились на персональном компьютере с видеокартой NVidia 3060Ti 8GB. Вычислены значения метрики качества — средней точности (Average Precision, AP) обнаружения объектов при пороговом значении пересечения по области (Intersection over Union, IoU) большем или равном 50%. Отдельно проанализированы скорость обработки изображений и количество памяти на видеокарте, которое нужно для развертывания модели на конечном устройстве.

Таблица 18 — Сравнение моделей сегментации объектов на тестовой выборке.

	AP(IoU=0.50), %	Скорость обр. изобр., кадр/сек	Потр. память, Гб
SOLov2 [582]	78.40	8	3.3
YolactEdge [653]	92.90	43	5.6

В целом, подход YolactEdge существенно превосходит нейросетевую модель SOLov2 как по быстродействию, так и по качеству. Но его существенным ограничением является более высокая потребляемая память графического процессора. Аппаратное обеспечение бортовой системы управления мобильного робота включает в себя видеокарту GTX1050Ti с 4Гб видеопамяти, которой не хватает для работы подхода YolactEdge. В результате при экспериментах на роботе использовалась модель SOLov2, которая, в целом, так же демонстрирует приемлемые показатели качества и быстродействия.

В ходе экспериментов также было произведено количественное сравнение двух подходов оценки трехмерной позы объекта: встроенного способа формирования карт глубин камеры Zed2 и способа на основе триангуляции координат боксов объектов на правом и левом изображениях. Мобильный робот был помещен в девять разных положений перед кнопкой, для которых были определены истинные значения 3D-позы с помощью манипулятора и известного фиксированного преобразования между базой манипулятора и камерой. Расстояние от камеры до кнопки составляло от 90 см до 110 см. Результаты сравнения приведены в табл. 19. Из нее видно, что способ на основе триангуляции дает более точную оценку трехмерных координат кнопки, чем встроенный метод формирования карт глубин камеры Zed2, но с большей неопределенностью (дисперсией) из-за ошибок в процедуре распознавания на двух изображениях. Стоит отметить, что такой способ получения 3D-позы сильно зависит от качества модели обнаружения объектов. Чтобы исключить возможность выбросов, был использован вероятностный фильтр на основе подсчета числа распознаваний в определенных областях. Следует отметить, что ключевая метрика для аккуратного позиционирования манипулятора по отношению к кнопке – дистанция до ее центра – определяется с помощью предложенного метода компьютерного

зрения со средней ошибкой 3 ± 2 мм. Это приемлемый показатель, свидетельствующий о работоспособности и практической полезности подхода.

Таблица 19 — Ошибка позиционирования кнопки, м.

	Средняя ошибка	Стандартное отклонение
Ошибка определения глубины (Zed2)	-0.017	0.003
Ошибка определения глубины (триангуляция)	-0.012	0.007
Дистанция до центра кнопки	0.003	0.002

Предложенная архитектура была использована для обеспечения мобильности робототехнической системы в офисном здании (учебном корпусе Физтех.Цифра Московского физико-технического института), оснащенном лифтами. Наиболее вычислительно сложными компонентами архитектуры являются связанные с обработкой визуальных данных картирование и распознавание кнопок. Эксперименты показали, что аппаратное обеспечение мобильного манипулятора не справляется с выполнением этих двух задач одновременно. С другой стороны, решать их одновременно не требуется, поэтому узел координации попеременно включает и выключает программные модули ответственные за навигацию и нажатие кнопок в зависимости от того, какая конкретно задача в данный момент выполняется.

В рамках сценария робот подъезжал к панели лифта, вызывал лифт, заезжал в него и выбирал нужный этаж. При заезде в лифт было решено не использовать планировщик, а управлять движением робота по заранее заданной траектории из точки перед внешней панелью в точку рядом с внутренней панелью. Управление колесной платформой оказалось достаточно точным, чтобы обеспечить решение данной задачи без использования планировщика в реальном времени. Схема координации ROS-узлов при выполнении сценария приведена на рисунке 5.9. Для наглядности процессов координации узлов эта схема организована несколько иначе, чем схема архитектуры на рисунке 5.2, и на ней опущены модули, отвечающие за реактивный уровень архитектуры (кроме модуля управления платформой). Стратегический узел-координатор (*lift-scenario-node*), ответственный за выполнение сценария, определяет, решение каких подзадач требуется при выполнении сценария. Эта информация передается на узел, ответственный за распределение ресурсов между процессами (*launcher-killer*), который предоставляет вычислительные ресурсы либо модулям локализации и построения карты (*Odometry*, *Navigation*) и планирования движения (*planner-cds*) колесной платформы, либо модулю распознавания объектов (*solo-node*). При заезде в лифт модуль управления платформой (*control*) отрабатывает предписанную траекторию, не используя данные навигации. Модуль планирования и управления манипулятором (*press-button*) не требует доступа к ресурсам, поскольку выполняется на выделенном аппаратном обеспечении манипулятора.

Пример выполнения сценария продемонстрирован последовательности кадров из видео, представленных на рисунке 5.10. Запись осуществлялась как камерой со стороны, так и камерой, закрепленной на рабочем органе манипулятора и используемой для распознавания

кнопок. На кадрах с этой камеры подсвечены распознанные маски кнопок на тех кадрах, для которых было включено распознавание.

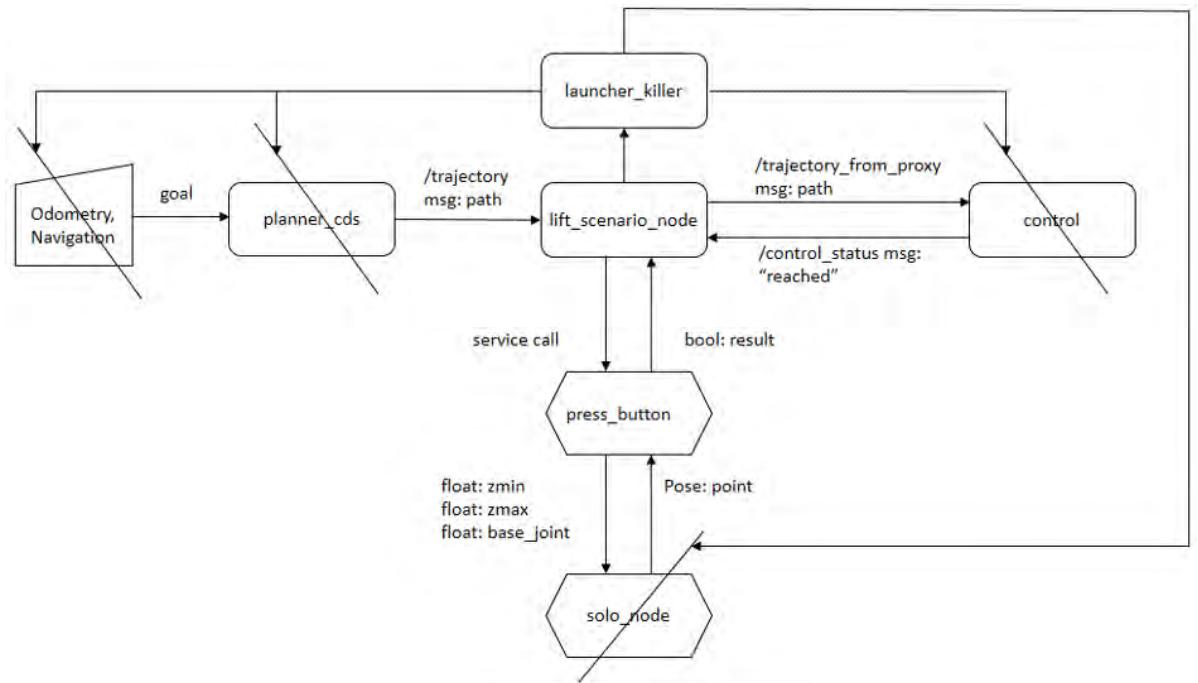


Рисунок 5.9 — Координация ROS-узлов при вызове лифта.

Использование колаборативных мобильных манипуляторов в сложной человеко-ориентированных средах имеет высокий практический потенциал и является областью активных исследований. В рамках представленных результатов данного раздела проведены адаптация и внедрение архитектуры NSLP и более общей архитектуры управления STRL. Также описана ее реализация для мобильных робототехнических платформ с манипулятором STRL-Robotics. В данной реализации была существенно расширена тактическая компонента архитектуры за счет добавления модулей управления манипулятором и построения сенсорной ситуации. Архитектура включает программные модули, которые можно разделить на две группы: обеспечивающие навигацию мобильного манипулятора на карте помещения и обеспечивающие взаимодействие манипулятора с объектами среды. В первой группе были разработаны и реализованы подходы к локализации и картированию, планированию движения и управлению движением колесной платформы. Во второй были реализованы методы и подходы для решения задачи использования лифта мобильным манипулятором, включающие алгоритмы распознавания и позиционирования кнопок лифта, а также подход к нажатию на кнопки лифта робототехническим манипулятором. Эффективность предложенных подходов подтверждается экспериментами на реальном роботе, в которых было показано успешное использование лифта. Предложенная архитектура управления является оригинальным объединением и синхронизированным набором оригинальных методов, реализующими ключевые подзадачи управления.

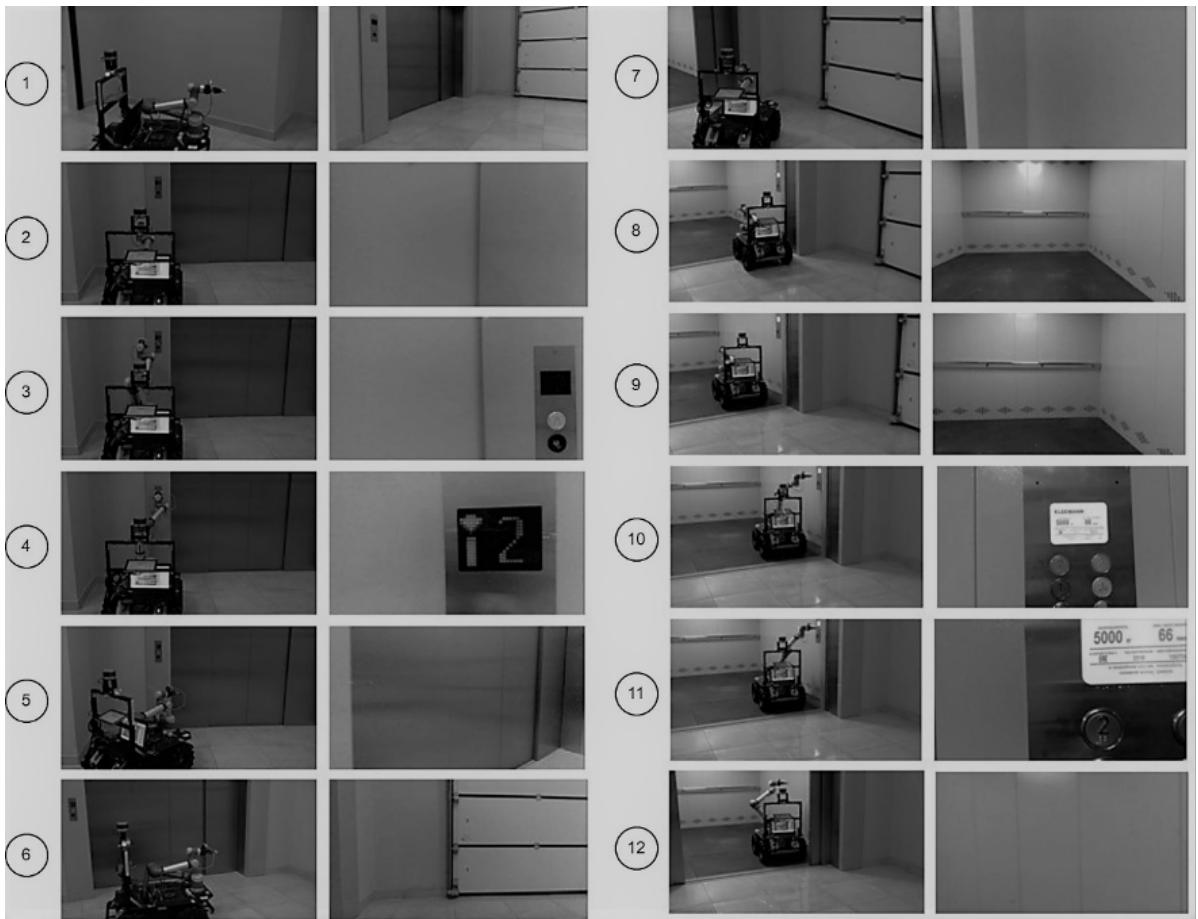


Рисунок 5.10 — Последовательность движений при вызове лифта.

5.2 Визуальная навигация робототехнической платформы в помещениях

В данном разделе диссертационного исследования рассмотрена интеграция методов классического планирования в качестве одного из вариантов умений и обучения с подкреплением в условиях архитектуры NSLP для задачи навигации в среде (рисунок 5.11). Рассмотрены подробности организации библиотеки умений κ_i и верхнеуровневой стратегии, которая реализует упрощенный вариант символьного планировщика с использованием обычного представления состояний без объектной декомпозиции. Основные результаты, изложенные в данном разделе, представлены в публикациях [9; 75].

5.2.1 Задача генерации маневров мобильного робота

В данном разделе рассматривается задача навигации когнитивного агента к объектам в замкнутых помещениях — одна из важнейших задач в воплощенном искусственном интеллекте, которая включает в себя генерацию действий агента для достижения объекта

целевого класса. Данную задачу под кратким названием *ObjectNav* можно разложить на несколько классических навыков автономной навигации, таких как навигация по координатам [206], исследование среды [393], быстрое перемещение [380] и достижение семантической цели [472], каждая из которых достаточно давно исследуется в мобильной робототехнике и в компьютерном зрении [557].

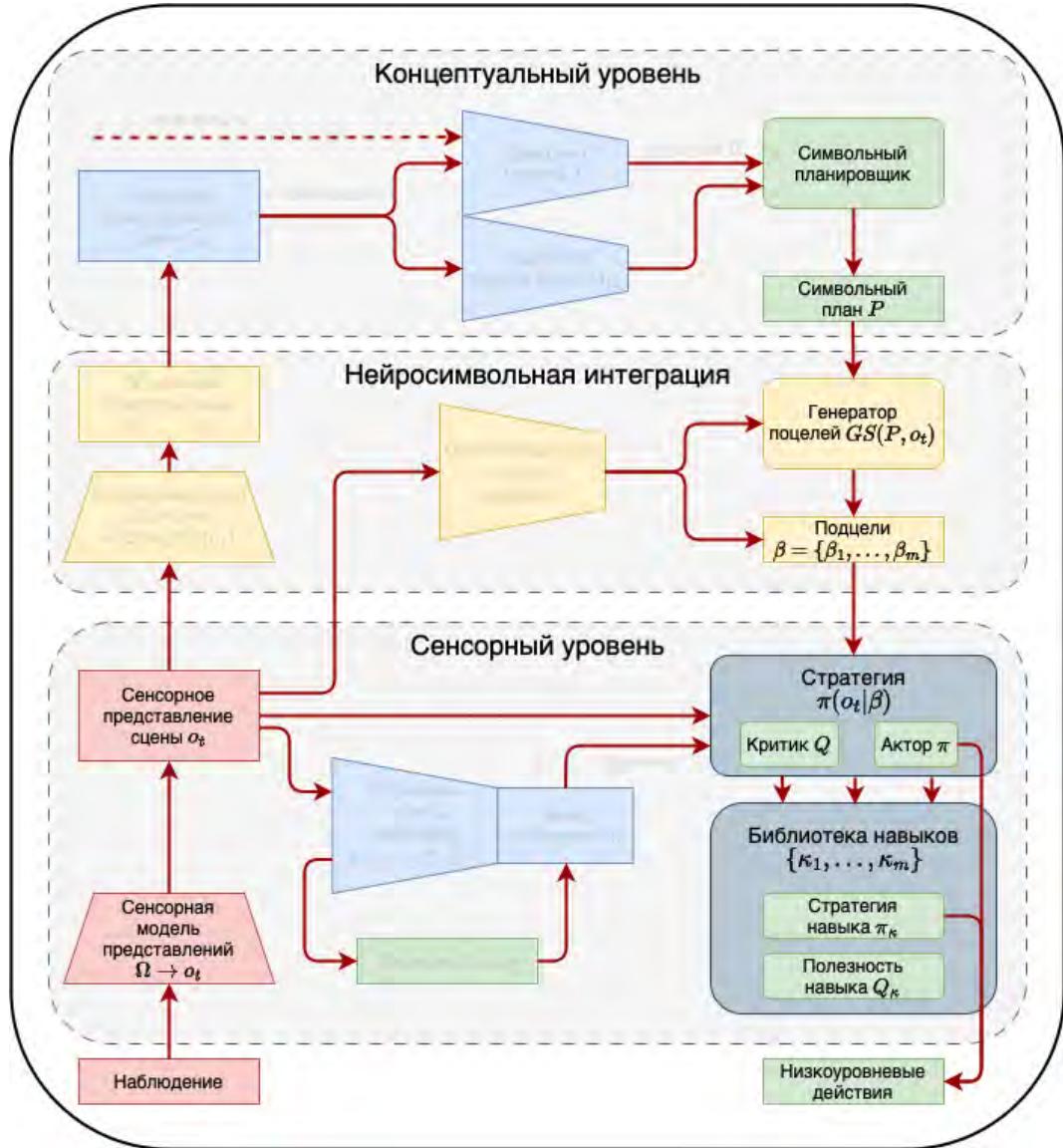


Рисунок 5.11 — Архитектура NSLP с выделенным циклом генерации действия по наблюдению и библиотекой навыков, включающей как обученные, так и классические стратегии.

Классические подходы к решению навигационных задач обычно включают в себя модульный процесс принятия решений по генерации действий, состоящий из компонентов, отвечающих за восприятие, картирование (карографирование), локализацию, планирование и собственно управление движением (реализацию конкретных действий по перемещению). Эффективность этих подходов зависит от множества факторов, таких как качество данных, поступающих от датчиков, сложность среды и количество используемых при конкретной реализации каждого компонента инженерных эвристик. Различные навигационные задачи

могут быть решены с помощью классических подходов как в симуляционных средах, так и на различных реальных роботизированных платформах [489; 569].

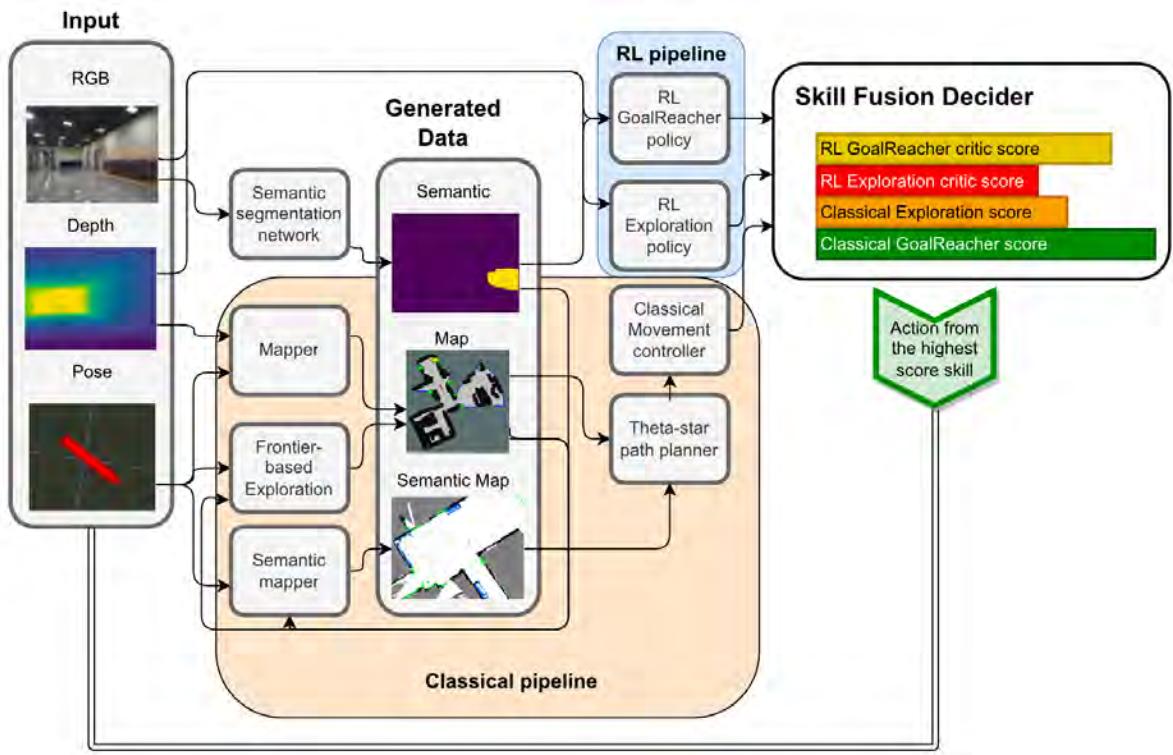


Рисунок 5.12 — Схема процесса принятия решений SkillFusion при решении задачи навигации к объектам, используемого на реальном роботе. Процесс состоит из классической части и части, основанной на обучении с подкреплением. Каждая часть обладает навыками исследования среды и достижения цели. Чтобы выбрать подходящий навык в каждый момент, реализуется решатель (decider) слияния навыков, который выбирает действие из набора навыков с наибольшей полезностью.

В последние годы широкое распространение получил отличный от модульного метод, основанный на так называемом сквозном (end-to-end) обучаемом подходе. Современные обучаемые методы позволяют агенту построить стратегии, реализующие не только свободное перемещение по сложным сценам без явной реконструкции карты [141; 206; 633], но и включающие возможности по манипулированию объектами [267], долгосрочному планированию [420], использующие обновляемые модели мира [419] и поддерживающие коммуникацию с людьми при помощи сообщений на естественном языке [606]. Успех этих методов обусловлен главным образом большим объемом данных, используемых для обучения.

В контексте реализации таких методов на робототехнических платформах для генерации данных обычно используются различные симуляторы, что приводит к снижению итоговой производительности методов в реальных условиях, которые могут отличаться от тех, на которых система была обучена. Другим недостатком таких методов является то, что, как правило, обучаемые методы полагаются только на неявное представление среды в памяти агента. На практике эти представления не вполне подходят для задач с долгосрочным планированием. В отличие от этого, классические, не использующие обучаемые компоненты

методы применяют явное представление окружающей среды посредством подходов по одновременной локализации и картированию (SLAM), и эти представления позволяют агенту осуществлять эффективное долгосрочное планирование в сложных сценах.

В данной реализации архитектуры NSLP предлагается преодолеть разрыв между классическим и обучаемым подходами за счет использования гибридной реализации одновременного обучения и планирования под названием SkillFusion, адаптированной к задаче ObjectNav (см. рисунок 5.12). Данный подход по гибридизации не является первым, где предлагается объединить обучаемые и необучаемые компоненты. Однако, в отличие от распространенных методов объединения (*fusion*), которые полагаются на модульные процессы принятия решений [468], предлагаемый подход учитывает специфические сильные стороны обеих парадигм и реализует динамическое объединение и реализацию решения, какой навигационный навык использовать, основываясь на оценке их полезностей. Эти навигационные навыки реализуются с использованием как классических методов, так и методов, основанных на обучении с подкреплением, позволяя агенту адаптивно выбирать наиболее подходящую парадигму решения задачи в процессе взаимодействия со средой.

Вклад проведенных исследований в рамках данного раздела в область работ по визуальной навигации заключается в следующем:

- Проведена декомпозиция задачи ObjectNav на два отдельных навыка: исследование среды, в котором агент должен эффективно обследовать различные области в среде в поисках целевого объекта, и навык достижения (GoalReacher), в котором агент должен переместиться к обнаруженному целевому объекту и остановиться рядом с ним. Была построена модульная архитектура SkillFusion, которая включает в себя как классическую, так и основанную на обучении реализацию каждого навыка. В процессе взаимодействия агента со средой в рамках эпизода агент принимает конкретные решения с использованием всех навыков в зависимости от их оценки полезности и внешних условий.
- С целью повышения устойчивости основанных на обучении стратегий агента к внешним условиям был использован предварительно обученный кодировщик визуального наблюдения и выбран набор навигационных задач для обучения, которые требуют от агента различного поведения, имея при этом одинаковые входные данные. Стратегии решения этих задач обучались одновременно с использованием техники раннего объединения [268], позволяющей итоговой стратегии использовать единый нейросетевой аппроксиматор с общими рекуррентными векторными представлениями для всех задач и отдельным выходом под каждую задачу.
- Чтобы решить проблему семантического шума (т.е. ложного обнаружения целевых объектов, реакции на выбросы и т.д.), был включен ряд техник как для необучаемых, так и для обучаемых навыков агента. Для необучаемых модулей были использованы техники эрозии и затухания для семантической карты. Для обучаемых было введено специальное действие «не уверен», которое помогает предотвратить направление

агента к ложно обнаруженному объекту путем повторного переключения на режим исследования среды для разрешения неопределенности.

Также необходимо отметить, что предложенный метод *SkillFusion* удалось перенести из симулятора в реальную среду без дообучения на предварительно собранных данных, как это делалось в других работах ранее [49]. Дополнительно были реализованы только более подходящие для реального робота регуляторы движения на уровне приводов.

Работы в области обучаемой навигации до целевых объектов

Обучаемые подходы к задаче навигации обычно подразумевают использование алгоритмов обучения с подкреплением (RL) для восстановления функции полезности или для непосредственного преобразования состояния в действие с использованием алгоритмов градиента стратегии. В этих алгоритмах для работы с сенсорными данными используются аппроксиматоры состояния, извлекающие из наблюдения признаковое представление, характерное для решения конкретной подзадачи.

Одним из подходов к получению более надежного и переносимого представления наблюдения является способ обучения аппроксиматора на множестве подзадач, то есть в режиме многозадачности. Этот подход хорошо подходит для рассматриваемой постановки задачи навигации с использованием нескольких навыков, поскольку он позволяет агенту отслеживать и запоминать различную информацию из одного и того же наблюдения, получаемого из среды [588]. Недавние работы, рассматривающие такой многозадачный подход, продемонстрировали возможность единого нейросетевого аппроксиматора эффективно кодировать наблюдение и способствовать успешному решению подзадач [104]. Проведенные эксперименты показывают, что совместное обучение нескольким навыкам методом раннего объединения [268] дает лучшие результаты по сравнению со сквозным обучением единой стратегии [206].

Другой подход к улучшению представления наблюдений в задаче навигации заключается в том, чтобы отделить визуальное кодирование наблюдения от аппроксимации функции стратегии. Этого можно достичь путем обучения кодировщиков на больших и более разнообразных, заранее собранных наборах данных, что позволяет получить кодировщик, не зависящий от конкретной решаемой задачи. Полученные в результате такого предобучения параметры кодировщика могут быть зафиксированы, что позволяет уже собственно алгоритму обучения стратегии использовать аппроксиматор с меньшим количеством параметров и быстрее сходиться [474; 636]. Также недавние исследования позволили установить, что использование предварительно обученных мультимодальных моделей, таких как CLIP [400], в качестве кодировщиков изображений в задачах навигации показывает высокую эффективность [510; 571].

Классический подход к навигации в ранее не наблюдавшихся сценах предполагает декомпозицию задачи навигации на следующие подзадачи: локализация, картирование,

планирование пути и следование по нему. Задачи локализации и картирования часто решаются совместно с использованием подхода одновременной локализации и картирования (SLAM). Большинство методов SLAM [483; 520; 597] извлекают признаковое описание сцены из наблюдений RGB-камеры или лидарных облаков и отслеживают движение робота, используя сопоставление этих признаков. Одним из самых популярных методов SLAM является RTAB-MAP [369]. Он способен работать с различными сенсорами (RGB-D камера, стереокамера, лидар) и создает как 2D-карту препятствий (occupancy grid), так и 3D-карту облаков точек.

Для планирования траектории движения чаще всего используются алгоритмы поиска путей на графе, таких как A* [311] или Theta* [617]. Граф обычно строят по клеточной карте препятствий, созданной с помощью одного из методов SLAM. Для следования по найденной траектории существует множество методов классического управления, которые обычно адаптированы к конкретной робототехнической платформе [327; 440; 579].

Другой важной задачей, связанной с навигацией к объектам внутри помещений, является исследование среды. Основным направлением в классической методологии решения задачи эффективного пространственного покрытия карты являются методы переднего края (frontiers) [127; 549]. Они выполняют поиск точек на границе областей, представленных свободными от препятствий и неисследованными клетками на клеточной карте препятствий, и выбирают одну из этих точек в качестве цели. Удобная и эффективная реализация алгоритма исследования на основе метода переднего края представлена в работе [456].

Важным направлением современных работ по части навигации до объектов является направление создания гибридных методов классической навигации и навигации, основанной на обучаемых подходах. В многочисленных работах показано, что когнитивные агенты, использующие обучаемые компоненты в своей архитектуре, хуже предотвращают столкновения с препятствиями и запоминают расположение объектов, но эффективноправляются с разрешением неоднозначности в данных и с шумами в сенсорах [361]. Одним из направлений объединения обучаемых и необучаемых подходов является разработка навигационных модулей на основе обучения с подкреплением, которые могут быть интегрированы в итоговый процесс принятия решения по движению. Алгоритм SLAM и планировщик движений могут быть представлены в виде параметризованных аппроксиматоров и обучены в рамках единой системы [186]. Для задач исследования среды и задачи ObjectGoal алгоритмы, представленные в работах [393] и [468], продемонстрировали превосходство модульных подходов над подходами, основанными на сквозном обучении и использующими только классические компоненты. В данных работах был задействован компонент, реализующий глобальную стратегию на основе обучения с подкреплением, который предсказывает точки переднего края, и компонент локальной стратегии, реализованной с помощью классического планировщика для навигации к этим точкам границы. В представленном подходе SkillFusion каждый навык не реализован как классическая или основанная на обучении функция генерации действий, а вместо этого является гибридным, то есть реализуется с использованием обоих подходов. Это

позволяет агенту определять, какой тип навыка следует выбирать в каждый момент времени, основываясь на функции полезности конкретного навыка.

Другим вариантом реализации гибридной стратегии принятия решений является так называемое байесовское объединение контроллеров (Bayesian Controller Fusion, BCF) [147], которое сочетает в себе стохастическую обучаемую стратегию управления с учетом неопределенности и контроллер, заданный на основе правил. Байесовская формулировка предполагает выбор того навыка, в данный момент реализующего управление, который демонстрирует наименьшую неопределенность в принятии решения. В условиях высокой неопределенности стратегия, реализованная на основе BCF, адаптирует распределение действий в сторону наименьшего риска, снижая таким образом вероятность катастрофического сбоя. Однако было обнаружено, что этот подход больше подходит для задач манипулирования с большими пространствами действий, где риск столкновения важнее метрик по решению конкретной задачи. С другой стороны, подход SkillFusion в большей степени учитывает итоговую отдачу по эпизоду взаимодействия со средой и в меньшей степени направлен на разрешение неопределенности в конкретный момент времени.

Постановка задачи навигации до целевого объекта

Задача навигации к целевому объекту внутри помещения (ObjectGoal) обычно определяется как задача перемещения к некоторому экземпляру заданного класса объектов $C \in \{c_1, c_2, \dots, c_n\}$ (например, к объекту класса *диван*) в ранее не наблюдаемой среде [472]. На каждом шаге агент получает наблюдение $S = (S_{RGBD}, S_{GPS+}, C)$. Пространство действий дискретно и состоит из четырех типов действий: **завершение работы** (callstop) для завершения текущего эпизода, **перемещение вперед** (forward) на 0,25м, **поворот влево** (turnleft) и **поворот вправо** (turnright) на угол $\alpha = 30^\circ$.

После окончания эпизода полученная траектория оценивается с помощью трех основных показателей:

- успешность достижения объекта: эпизод считается успешным, если агент выполняет команду callstop в пределах 1,0м от любого объекта целевого типа;
- успешность достижения объекта, взвешенная по нормализованной длине пути (SPL), что служит показателем эффективности найденного агентом пути к ближайшему объекту от начальной точки;
- метрика SoftSPL, где бинарная оценка достижения или недостижения объекта заменяется на евклидово расстояние до цели.

Сформулированная задача ObjectGoal является сложной задачей, требующей от агента реализации адаптивного поведения. Это поведение может быть декомпозировано на отдельные навыки агента. Были выделены два основных навыка, которые непосредственно решают задачу ObjectGoal: навык исследования (Exploration skill) и навык достижения цели (GoalReacher skill) — и два дополнительных: навык навигации по координатам (PointNav

skill) и навык быстрого перемещения (Flee skill). Для каждого из основных навыков были реализованы как классические, так и основанные на обучении решатели. Дополнительные навыки используются для улучшения обобщающей способности обучаемой стратегии.

- Целью навыка исследования (Exploration skill) является эффективное исследование как можно большей области пространства среды за ограниченный промежуток времени.
- Навык достижения цели (GoalReacher skill) имеет целью направить агента к заданному объекту, который был выделен в текущем наблюдении, и должен завершаться выполнением действия `callstop` в пределах 1,0м от этого объекта.
- Цель навыка навигации по координатам (PointNav skill) — направить агента в заданную координатами точку пространства и выполнить действие `callstop` на расстоянии 1,0 м от нее.
- Цель навыка быстрого перемещения (Flee skill) — выполнить действие `callstop` в самой удаленной точке от начальной позиции агента.

Конкретное воплощение агента в симуляторе, реализующем взаимодействие агента со средой, реализовано с использованием модели робота LoCoBot (недорогой мобильной робототехнической платформы). Размер основной платформы робота составляет 0,18м. Определение истинного положения робота (локализация) обеспечивается с помощью комплексного сенсора *GPS+Compass* (генерирующего часть наблюдения $S_{GPS+Compass}$). Выполнение действия предполагается детерминированным, без случайного шума, но координаты целевых объектов и истинная семантическая сегментация недоступны агенту.

В проведенных в данном разделе работы экспериментах с реальным воплощением агента использовалась платформа Clearpath Husky с дифференциальным приводом на четыре колеса, размеры которой составляют $990 \times 670 \times 390$ мм. Несмотря на то что эти параметры существенно отличаются от варианта воплощения в симуляторе, было показано, что метод SkillFusion способен легко адаптироваться к конкретной роботизированной платформе без специальной настройки стратегии путем дообучения и изменения других компонентов архитектуры.

В симуляторе агент получает точные наблюдения по монокулярной камере RGB-D, установленной на высоте 0,88 м. Разрешение камеры составляет 480×640 пикселей при горизонтальном поле зрения в 90° . Дальность датчика глубины составляет 5 метров.

Для соответствия этим входным сигналам в реальной среде платформа Husky была оснащена лидаром Velodyne VLP-16 и RGB-D камерой. Дальность действия всех сенсоров была ограничена 5 метрами. Лидар используется для точной локализации и построения карты препятствий, чтобы компенсировать отсутствие датчика GPS+компаса в симуляторе.

5.2.2 Классические навыки генерации действий

Предлагаемая реализация классического навыка исследования среды состоит из следующих модулей: генерация цели, локализация, картирование, планирование траектории и управление движением.

Модуль постановки целей использует текущую оценку положения робототехнической платформы, построенную с помощью SLAM-карты, и выбирает цель, к которой агент должен двигаться. Если целевой объект нанесен на карту, предоставляемую методом SLAM, навык навигации выбирает ближайшую к нему точку.

В качестве модуля постановки целей была использована реализация подхода к исследованию на основе метода переднего края, который был предложен в работе [456]. Данный метод ищет границу между исследованным и неизвестным пространством на 2D-карте, построенной с помощью SLAM. Для реализации поиска точек границы используется стандартный поиск в ширину (breadth-first search, BFS).

Всем точкам границы присваиваются значения стоимости, а центроид для данного множества точек границы, построенный по наименьшему значению функции стоимости, помечается как цель для робота. Функция стоимости точек границы учитывает расстояние между агентом и центроидом, количество точек границы и угол, на который агент должен повернуться, прежде чем он начнет двигаться к точке границе. Ниже данная процедура описана формально.

Пусть $q \in \mathbb{R}^2$ — вектор ориентации робота, а $\pi = (p_0, p_1, \dots, p_k)$ — путь от агента к центроиду i -го множества точек границы размерностью n_i . Пусть π_{p_0} — это начальная позиция агента, а p_k — центроид i -го множества точек границы. Используемая функция стоимости выглядит следующим образом:

$$cost_i = \alpha \sum_{j=0}^k \|p_{j+1} - p_j\|_2 - \beta n_i + \gamma |\angle(q, p_1 - p_0)|, \quad (5.5)$$

где α — коэффициент для учета длины пути, β — коэффициент для учета количества точек, а γ — коэффициент для учета угла поворота между ориентацией робота и направлением на точку границы.

В симуляции используются истинные данные о точном местоположении, поэтому реализация метода локализации не требуется. Чтобы максимально повысить точность локализации и избежать накопления ошибок, на реальном роботе был использован алгоритм лидарной одометрии LOL-odom [539] с дополнительной коррекцией позы с помощью предварительно созданной 3D-карты (этот карта используется только для коррекции позы).

Модуль картирования создает клеточную карту препятствий для навигации на этапе исследования среды и семантическую карту для навигации к целевому объекту. В симуляционной среде, где доступны истинные данные об объектах среды, была использована обратная проекция карты глубины для построения карты препятствий. Для построения семантической карты была применена обратная проекция прогнозируемых семантических

масок объектов. Все объекты высотой менее 0,2 м отображаются как свободные клетки, а все объекты высотой более 0,2 м помечаются как препятствия. На реальном роботе был использован алгоритм RTAB-MAP SLAM [369] с лидаром в качестве источника облака точек. Семантическая информация добавляется к карте RTAB-MAP из семантических масок, прогнозируемых с помощью камеры RGBD.

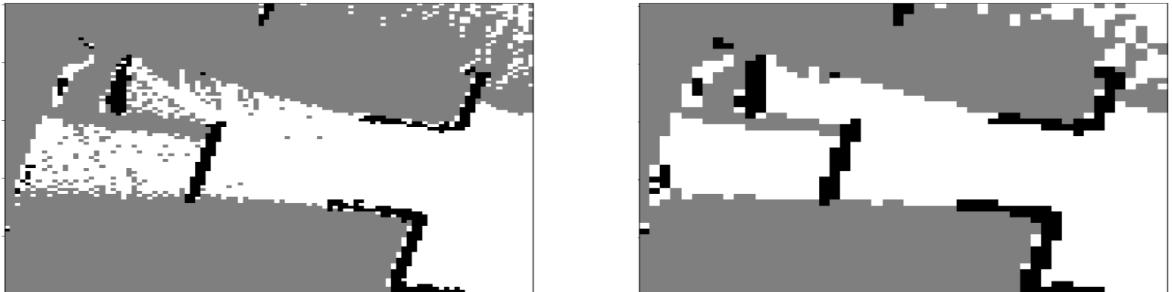


Рисунок 5.13 — Пример объединения по максимуму построенной с помощью SLAM карты. Серым цветом обозначена неизвестная область, черным — клетки с препятствиями, а белым — клетки свободного пространства.

Как в симуляторе, так и на реальном роботе формируемая клеточная карта имеет разрешение 5 см. Чтобы закрыть небольшие «дыры» в карте и уменьшить количество избыточных границ, появляющихся из-за неточности выделения объектов на карте, размер клетки увеличивается в два раза с помощью метода объединения по максимуму (max pooling) с порядком значений клеток «неизвестно < свободно < препятствие». Таким образом карта разбивается на квадраты 2×2 , и каждый квадрат образует единственную клетку на карте уменьшенного размера. Эта клетка помечается как препятствие, если квадрат содержит хотя бы одну клетку с препятствием. Если квадрат 2×2 содержит только клетки свободного пространства и в ней отсутствуют клетки с препятствиями, результирующая клетка на карте измененного размера помечается свободной. Если квадрат 2×2 содержит только клетки с неизвестным значением занятости, результирующая клетка на карте измененного размера помечается как неопределенная. Пример такого объединения по максимуму показан на рисунке 5.13.

Чтобы предотвратить выбросы в прогнозируемых семантических масках при реализации функции картирования, были реализованы техники эрозии семантической карты и накопления семантической информации. Была использована эрозия (размытие) целевых объектов на семантической карте на две клетки. Также были использованы непрерывные значения вероятности занятости клеток семантической карты вместо двоичных значений. Если семантический объект проецируется на клетку карты, значение в этой клетке увеличивается на 1 — в противном случае значение умножается на коэффициент $\alpha < 1$. Клетка помечается как клетка целевого объекта, если ее значение превышает некоторое пороговое значение T . В проведенных экспериментах были использованы гиперпараметры $\alpha = 0.9$ и $T = 2$.

Для планирования траектории агента был использован алгоритм Theta* [617]. Это алгоритм поиска пути в клеточных средах, который поддерживает перемещение агента под

любым углом. По умолчанию Theta* планирует путь для агента нулевого размера, поэтому этот алгоритм был модифицирован для учета ненулевого размера робота. В используемой модификации агент моделируется как круг радиусом r . На практике был установлен размер круга больше фактической площади робота, чтобы обеспечить (дополнительный) запас расстояния до препятствий. Если путь не был найден, то безопасный радиус до $0,8r$ и алгоритм планирования пути запускались заново. В экспериментах устанавливались следующие значения: $r = 0,15$ в симуляции и $r = 0,6$ для робототехнической реализации.

Для следования построенному пути в симуляционной среде был использован упрощенный вариант регулятора на правилах. Сравнивались ориентация робота и направление от текущего положения робота к следующей точке траектории. Если угол между ориентацией робота и направлением к точке траектории меньше некоторого порогового значения (например, менее 5° по абсолютной величине), то робот перемещается вперед. Если он выше порогового значения и имеет отрицательное значение, то робот поворачивается влево. Если он выше порогового значения и имеет положительное значение, то робот поворачивается вправо.

На реальном роботе используется реализация метода частичного отслеживания траектории⁷, который получает на вход требуемую траекторию и положение робота и генерирует конкретные значения скорости для робота. Этот подход к реализации следования по траектории похож на основанный на правилах подход, реализованный для SkillFusion в симуляционной среде, но адаптирован для случая непрерывных действий, реализующих движение, и имеет некоторые эвристические возможности для компенсации ошибок одометрии.

Для реализации навыка достижения цели были использованы те же модули локализации, планирования, картирования и контроллера движения, что и для навыка исследования среды, описанные выше. Для генерации целей снова применялся подход, основанный на методе переднего края, но вместо точек границы между известными и неисследованными областями карты использовались точки области вокруг целевых объектов.

5.2.3 Обучаемые навыки генерации действий

Как и реализация классических навыков в архитектуре SkillFusion, принятие решения на основе обученных навыков опирается на два компонента: стратегию исследования среды и стратегию достижения целей. Чтобы обучить оба эти навыка, был использован алгоритм децентрализованной распределенной оптимизации ближайшей стратегии (DD-PPO), поскольку он показывает высокие результаты в аналогичных рассматриваемой задачах визуальной навигации [206]. Архитектура нейросетевого аппроксиматора для обучения

⁷https://github.com/andrey1908/str1/_robotics/tree/main/control

предлагаемой стратегии показана на рисунке 5.14. Для получения высоких результатов требуется огромное количество этапов обучения. При этом необходимо использовать быстродействующий симулятор, который также должен быть фотoreалистичным, чтобы иметь возможность перенести полученную стратегию генерации действий в реальную среду. В связи с этим для обучения был использован наиболее быстрый в настоящий момент фотoreалистичный симулятор BPS [380] с большим набором данных из 1000 сцен — HM3D [306]. Для обучения навыка исследования среды была использована обучающая выборка из набора HM3D (800 сцен), а для обучения достижения цели — только 145 сцен, поскольку в HM3D для этого числа сцена имеется истинная информация о семантических масках объектов.

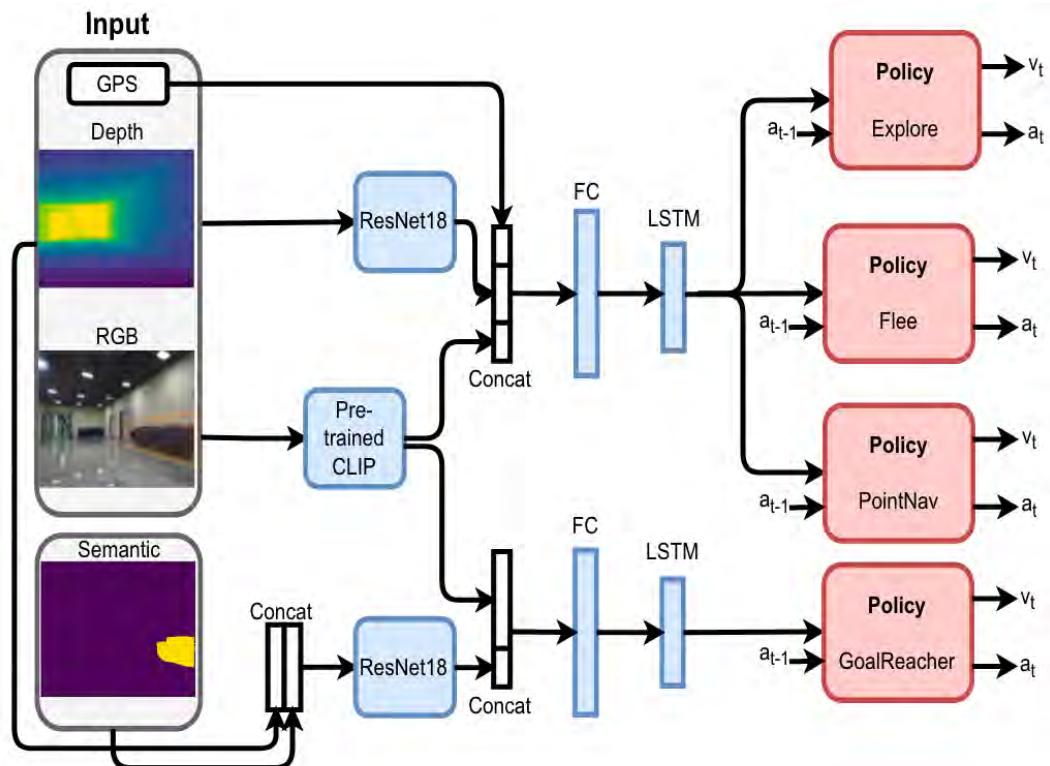


Рисунок 5.14 — Предлагаемая архитектура нейросетевого аппроксиматора для совместного обучения навигационных навыков.

Навык исследования среды

Исследование среды — сложная задача для обучаемых подходов, требующая точности в выполнении действий, способности запоминать прошлый опыт и планировать будущие перемещения. В реальных сценах и в фотoreалистичных средах эта задача усложняется значительными различиями в размерах помещений, планировке, наличии различных объектов, их цвете, текстуре и других параметрах. Чтобы алгоритм обучения мог обобщить такую разнородную информацию, для него необходим высокопроизводительный

кодировщик наблюдений для извлечения всех необходимых ключевых признаков по данным зашумленного сенсора RGB. Данная проблема была решена путем использования двух частей кодировщика: надежного кодировщика по обработке визуальных наблюдений в виде изображений и RNN-кодировщика, запоминающего динамические закономерности в наблюдениях.

В качестве кодировщика изображений была использована предварительно обученная мультимодальная модель CLIP [400], параметры которой были зафиксированы на этапе обучения. Этот подход применялся и в предыдущих работах по навигации [510; 571], и проведенные эксперименты демонстрируют возможность реализации навыка навигации только по данным RGB сенсора с таким зафиксированным кодировщиком на базе модели CLIP.

Для реализации динамического надежного RNN-кодера все навигационные навыки были объединены в общую архитектуру в стиле раннего объединения (early fusion) [268] (см. рисунок 5.14). Обучение проводилось с использованием нейросетевого аппроксиматора с общими слоями базового визуального кодировщика и рекуррентными слоями (RNN) и несколькими выходами в соответствии с рассматриваемыми навыками. Навык исследования среды должен обобщать пространственную структуру среды, чтобы избегать попадания в уже исследованные области, в то время как навык быстрого перемещения фокусируется на оценке расстояния и ориентации. Навык навигации по координатам в основном сосредоточен на поиске кратчайшего возможного пути к точке. Специализированная функция вознаграждения в каждом подзадаче опосредует такое поведение:

- Вознаграждение для навыка исследования среды устанавливается равным +1, если агент посетил ранее неисследованную клетку в размером 1 m^2 , в противном случае 0.
- Вознаграждение за быстрого перемещения пропорционально расстоянию от начальной точки.
- Вознаграждение за навык навигации по координатам пропорционально сокращенному расстоянию до цели, плюс предусматривается дополнительное вознаграждение, если агент выполняет действие остановки в пределах 1 м от целевой точки.

Навык достижения целевого объекта

Важнейшим аспектом при обучении навыка достижения цели является способность отличать целевые объекты от объектов, не являющихся целью. Чтобы включить более полную информацию об объекте в наблюдение агента, вместо его идентификатора была использована бинарная маска сегментации этого объекта. Этот подход позволяет обучить модель семантической сегментации независимо от остальных компонент системы. Недостатком подхода является необходимость обучать стратегию поведения с помощью

истинного семантического сенсора, который служит для генерации предусмотренного вознаграждения. Важно отметить, что в валидационных эпизодах обученная модель семантической сегментации выдает большое количество шумов и ложноположительных объектов, которые могут появляться в одном кадре, но исчезать в следующем.

Навык достижения цели обучается с большим положительным вознаграждением за успешное завершение эпизода и небольшим отрицательным вознаграждением за каждый совершенный в среде шаг. В результате появляющиеся из-за неточности сегментатора шумовые объекты могут привести к тому, что агент ошибочно завершает эпизод в месте, где появляется шумовой объект, что, в свою очередь, приводит к большому отрицательному вознаграждению. Другой тип ошибочного поведения заключается в том, что агент может попытаться достигнуть удаленного шумового объекта и накопить большое значение суммы небольших отрицательных вознаграждений, выполнив слишком много действий.

В связи с этим, чтобы позволить агенту отличать шумовые объекты от действительно присутствующих на сцене и позволить ему избежать реализации ошибочного поведения в неблагоприятных ситуациях принятия решений, было предложено использовать специальное действие «не уверен». Когда степень неопределенности агента в достоверности маски сегментации высока, он может выполнить это действие, чтобы вернуться к навыку исследования среды, вместо того чтобы продолжать реализовывать навык достижения шумового объекта.

На этапе обучения навыка достижения целевого объекта были отобраны эпизоды, в которых с вероятностью 0,2 проявлялась семантическая сегментация шумовых объектов. Остальные эпизоды содержали истинную базовую сегментацию. В шумовых эпизодах агент получает промежуточное отрицательное вознаграждение, если он выбирает действие «не уверен» при завершении эпизода, и большое отрицательное вознаграждение, если он выполняет действие «стоп». Цель агента состоит в том, чтобы быстро распознать, является ли семантическая сегментация ошибочной, и завершить эпизод действием «не уверен» или продолжить достигать цель в случае уверенности в точности сегментации и выполнить действие «стоп» рядом с целевой позицией.

5.2.4 Гибридная верхнеуровневая стратегия выбора навыков

Для эффективного использования преимуществ как классического подхода, так и подхода, основанного на обучении, был предложен метод объединения перечисленных выше навыков. Как уже было указано, по отдельности каждый из этих навыков имеет ряд существенных ограничений.

Классические методы исследования среды, основанные на методе переднего края, реализованного на 2D-карте, надежны и устойчивы к изменениям сцены и положения камеры. Однако они требуют наличия точной одометрии и данных с метода картирования. С другой стороны, подходы, основанные на обучении, могут быть более эффективными

при решении подзадач навигации на конкретных типах сцен, поскольку их можно обучить решать эти подзадачи напрямую, в том числе и при задании специализированной функции вознаграждения. В навигации к целевому объекту подходы, основанные на обучении, показывают большую эффективность в сценах, где цель может быть найдена за короткое время, но на больших расстояниях рекуррентный нейросетевой аппроксиматор начинает забывать информацию с начальных этапов эпизода и совершать обходы ранее посещенных мест. Напротив, классические навыки хранят всю информацию с самого начала в виде 2D-карты и могут исследовать новые места в течение всего времени работы в рамках эпизода взаимодействия агента со средой. Кроме того, подходы на основе обучения с подкреплением показывают значимые результаты в поиске цели и ее достижении, но для них оказывается сложным генерировать действия остановки вблизи объекта.

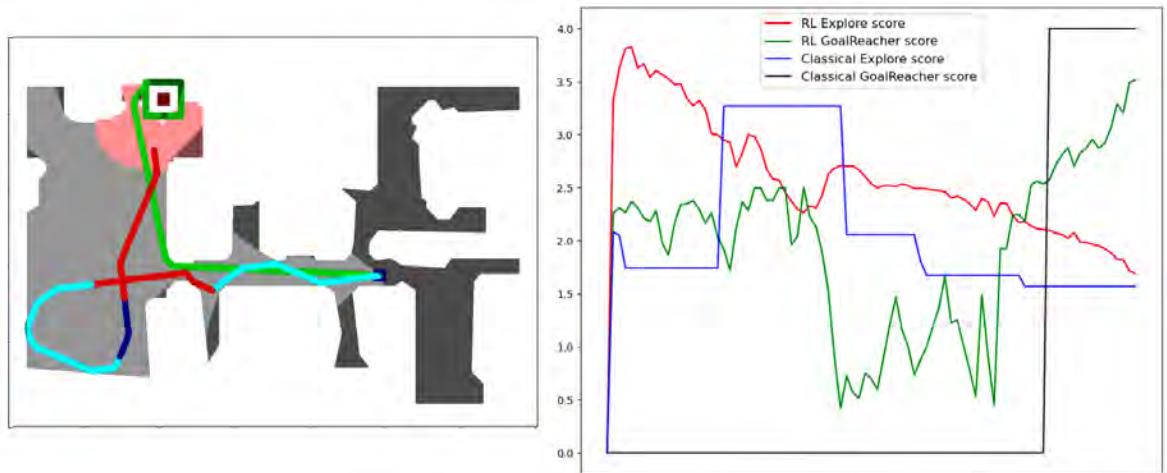


Рисунок 5.15 — Пример процесса выбора навыков в рамках одного эпизода. Синий цвет траектории обозначает выполнение обучаемого навыка исследования среды, темно-синий обозначает обучаемый навык достижения цели, а красный цвет обозначает классические навыки.

Чтобы эффективно использовать преимущества каждого подхода, в архитектуре SkillFusion реализуется модуль, получающий информацию о преимуществах, которые имеющиеся навыки могут предоставить в каждый момент времени. Для этой цели возможно использование специальной функции стоимости: для навыков, обучающихся в режиме «актор-критик», это функция полезности критика, а для классических навыков — это функция стоимости для точек границы метода переднего края (см. рисунок 5.15). Была проведена нормализация значений этих функций в одном и том же масштабе в соответствии с данными, составляющими накопленный агентом опыт в рамках нескольких эпизодов. На каждом шаге агент выполняет навык с максимальным значением данной функции стоимости. Поскольку целью задачи ObjectNav является выполнение действия остановки, когда агент находится рядом с целью, классическому навыку, реализующему навык достижения цели, присваивалось постоянное наивысшее значение в тех случаях, когда цель появляется на карте после проведенной фильтрации.



Рисунок 5.16 — Примеры работы верхнеуровневой стратегии в симуляционной среде. Красная линия — это траектория, выполненная с классическим навыком, светло-синяя линия — с обучаемым навыком исследования среды, а темно-синяя — с обучаемым навыком достижения цели.

В результате использования такой высокоуровневой стратегии агент начинает исследование среды с использованием обучаемого навыка, чтобы эффективно охватить большие площади вокруг начальной позиции. Через некоторое время обучаемый навык из-за неэффективной реализации рекуррентных слоев забывает информацию и начинает исследовать уже посещенные области. Когда это происходит, оценка этого навыка становится ниже оценки классического варианта и агент переключается на классический навык. В этот момент классической навык исследования среды, основываясь на информации о текущей карте, перемещает агента к границе с наибольшим значение функции стоимости. После этого оценка обучаемого навыка снова становится выше оценки классического, и верхнеуровневая стратегия переключается обратно на обучаемую реализацию (см. рисунок 5.16 слева). Необходимо отметить, что слои рекуррентные слои обучаемых навыков обновляются на каждом шаге, даже когда агент выполняет классические навыки.

Если объект цели в результате работы семантической сегментации попадает в наблюдение агента, оценка навыка достижения цели существенно увеличивается и верхнеуровневая стратегия переключается на этот навык, пока объект не исчезнет с семантической карты. При использовании обучаемого навыка достижения цели у агента есть два варианта: завершить эпизод самостоятельно, выполнив действие «стоп» (см. рисунок 5.16 справа), или вернуться к выполнению навыков исследования среды, выполнив действие «не уверен», если он решит, что объект цели является шумом. Когда целевой объект находится на карте сегментации, оценка классического навыка достижения цели устанавливается на постоянное максимальное значение, и агент автоматически переключается на его выполнение (см. рисунок 5.16 посередине).

Таблица 20 — Метрики качества метода SkillFusion по сравнению с базовыми алгоритмами на наборе данных HM3D.

Method	Success	SPL	SoftSPL
DDPPO [206]	0.18	0.10	0.35
SemExp [468]	0.24	0.14	0.26
Auxiliary RL [141]	0.51	0.29	0.34
SkillFusion	0.64	0.36	0.38

5.2.5 Экспериментальное исследование метода SkillFusion

Эксперименты в симуляционной среде Habitat

В табл. 20 представлены результаты работы метода SkillFusion в сравнении с современными базовыми алгоритмами в симуляционной среде Habitat [307]. В качестве валидационных эпизодов были использованы 20 сцен из выборки Val набора данных HM3D [306], которых агент не наблюдал на этапе обучения. Итоговое количество эпизодов равно 120, с равномерным распределением классов целевых объектов (стул, кровать, растение, монитор, диван и др.) и средним минимальным расстоянием до ближайшего объекта 8 м. В качестве базовых методов для сравнения были выбраны следующих три метода. Алгоритм DDPPO [206] — это метод сквозного обучения с подкреплением, предоставленный разработчиками симулятора Habitat. Алгоритм SemExp [468] — это модульный подход, который включает в себя объединение глобального обучаемого планировщика и классического локального планировщика. Алгоритм Auxiliary RL [141] — это подход на основе обучения с подкреплением, дополненный вспомогательными функциями потерь. Для всех базовых методов были выбраны доступные открытые реализации и использованы полученные авторами методов веса аппроксиматоров. Для того, чтобы эти методы можно было применить к набору данных HM3D и целевому типу объектов, вместо базовых модулей семантической сегментации были использованы истинные данные из среды.

RGB-кодировщик является существенной частью нейросетевого аппроксиматора во всех обучаемых подходах, замедляющей весь процесс обучения. Кроме того, в наборе данных для обучения имеется лишь ограниченное количество сцен, в то время как предполагается, что агент будет перемещаться в ранее не наблюдавшихся сценах. Следовательно, сохранение высокого уровня надежности RGB-кодировщика для новых сцен является сложной задачей. Для решения этой проблемы может быть использован предварительно обученный RGB-кодировщик, который фиксируется на этапе обучения стратегии. Для реализации этого подхода был выбран кодировщик на базе модели CLIP [400]. Чтобы продемонстрировать, что уменьшение обучаемых параметров по-прежнему позволяет агенту выполнять все

подзадачи навигации, было проведено сравнение производительности в задаче достижения цели с зафиксированной моделью CLIP с обучаемым кодировщиком на базе модели ResNet 5.17. Как показывает проведенный эксперимент, производительность агента с зафиксированной моделью не ухудшается. Чтобы проанализировать зависимость полученной стратегии от использования данных глубины и показать, что нейросетевой аппроксиматор эффективно использует CLIP представления не только для распознавания объектов, но и для решения навигационных задач, был обучен навык достижения цели только на RGB-данных. Метод SkillFusion в данной реализации по-прежнему работает с высокой производительностью, однако все-таки уступает полной архитектуре.

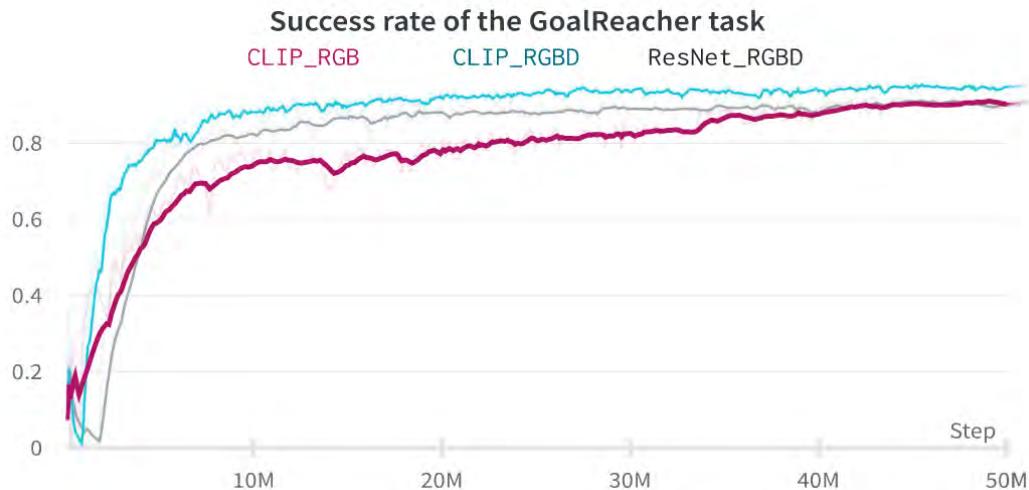


Рисунок 5.17 — Сравнение кривых обучения навыка достижения цели с использованием зафиксированного кодировщика на базе модели CLIP и с обучаемым кодировщиком на базе модели ResNet.

С целью демонстрации необходимости объединения классических и обучаемых навыков, были протестированы различные комбинации этих навыков при решении общей задачи навигации. Результаты сравнения показывают, что только обучаемый подход показывает лучшие результаты, чем только классический подход по метрике SPL в симуляционной среде, но имеет сопоставимый показатель успешности SR (см. табл. 21). Этот факт объясняется тем, что классический процесс принятия решений полагается на точную карту и оказывается «слепым» за пределами диапазона работы датчика глубины (5 м), что приводит к тому, что в этом случае агент тратит много времени на изучение различных локальных тупиковых зон. В то же время обучаемый подход может прогнозировать эту карту в неявном виде, извлекая информацию из RGB-сенсора. Сопоставимый показатель успеха SR обусловлен тем фактом, что в рамках классического метода проще определить момент, когда агенту необходимо завершить эпизод при обнаружении целевого объекта, поскольку в этом подходе можно надежнее рассчитать расстояние до объекта с помощью точной карты препятствий.

Сочетание классических и основанных на обучении методов при решении задачи исследования среды повышает итоговую эффективность стратегии агента. В результате он находит больше целевых объектов, одновременно увеличивая показатель SPL и показатели

успешности завершения эпизода. Наконец, добавление обучаемого навыка достижения цели увеличивает количество успешных эпизодов на заключительных шагах траекторий, что также положительно отражается на всех показателях.

Таблица 21 — Исследование влияния различных навыков на итоговую эффективность метода SkillFusion в рамках одного эпизода.

Skill		Metrics		
Explore	GoalReacher	Success	SPL	SoftSPL
Classical	Classical	0.410	0.182	0.263
RL	RL	0.403	0.224	0.321
RL+Classical	Classical	0.511	0.299	0.309
RL+Classical	RL+Classical	0.547	0.316	0.365

Как показано в табл. 22, производительность метода SkillFusion чувствительна к качеству семантического сенсора. С истинной семантикой (GT) результаты почти вдвое превосходят результаты с использованием сегментатора на базе метода SegFormer. Однако с реализацией техник фильтрации семантической карты для классических навыков и с добавлением действия «не уверен» в обучаемые навыки этот разрыв практически компенсируется.

Таблица 22 — Исследование влияния модуля семантической сегментации на метрики метода SkillFusion.

Method	Success	SPL	SoftSPL
SkillFusion (with no semantic filtering)	0.324	0.207	0.327
SkillFusion (with semantic filtering)	0.547	0.316	0.365
SkillFusion (ground truth semantic)	0.647	0.363	0.384

Эксперименты на робототехнической платформе Habitat

Помимо симуляционных экспериментов, был проведен ряд тестов на роботе Clearpath Husky (см. рисунок 5.18). Перед роботом также ставилась задача по поиску различных объектов (например, стул или диван) в неизвестном помещении (здание университета). Он выполнял задачи без какой-либо дополнительной настройки обученной в симуляторе стратегии.

Для того чтобы выявить сильные и слабые стороны классического и обучаемого методов принятия решений в случае с реальным роботом, на первом этапе они были протестированы отдельно, а затем оба подхода были сравнены с полным методом SkillFusion. Все варианты были запущены в семи сценах с тремя различными целевыми объектами: синим стулом, красным стулом и синим диваном. Синий стул был расположен в углу

коридора. В тесте с диваном два больших синих дивана были расположены в разных углах коридора. Красное кресло располагалось за узким проходом в коридоре, и робот должен был исследовать весь коридор, прежде чем войти в этот узкий проход, чтобы достичь цели. Для синего кресла и диванов все подходы были запущены из трех разных исходных местоположений, в то время как для красного кресла было использовано только одно исходное местоположение.



Рисунок 5.18 — Роботизированная платформа на базе Clearpath Husky с камерой ZED (слева). Полигон, который был использован для оценки результатов в реальных условиях (справа).

Траектории для всех подходов показаны на рисунке 5.19, а значения метрик приведены в табл. 23. Были измерены пять показателей: успешность решения задачи (SR), метрики SPL и SoftSPL, длина пути, пройденного роботом, и время прохождения пути. При использовании обучаемого подхода робот дважды не смог достичь целевого объекта. В тесте с синим стулом робот добрался до дивана вместо стула. Это произошло вследствие ошибок семантического сегментатора и упрощенной реализации навыка достижения цели без семантической фильтрации данных. В тесте с красным стулом робот проигнорировал узкий проход, поскольку обучаемый навык был обучен с функцией вознаграждения, пропорциональной исследованной области. В результате оценка полезности действий по попаданию в узкий проход была слишком низкой.

В одном teste из трех роботов не смог найти синий стул, используя классический метод (см. рисунок 5.19). Это было связано с ограниченным диапазоном в 5 м для семантического картирования и шумов семантического предсказания, из-за чего целевой стул игнорировался. В результате робот исследовал весь коридор, не отметив стул в качестве цели.

Используя полный метод SkillFusion, робот успешно прошел все тесты, достигнув среднего значения SPL и SoftSPL 0,65. Среднее расстояние перемещения было сокращено до 23 м по сравнению с 27 м при классическом подходе и 31 м при обучаемом подходе. Этот результат демонстрирует успешный перенос обученной в симуляторе стратегии и успешное сочетание сильных сторон как классического, так и основанного на обучении подходов при реализации на реальном роботе.

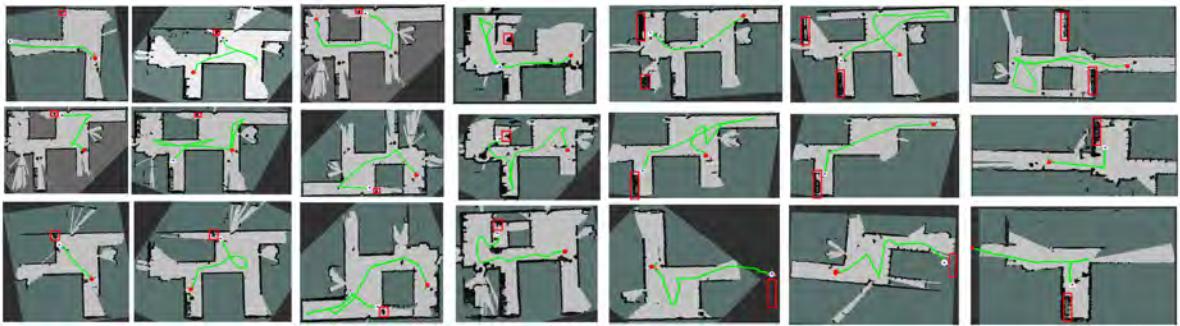


Рисунок 5.19 — Траектории движения реального робота с разными подходами: обучаемым (вверху), классическим (посередине) и полным методом SkillFusion (внизу). Красный прямоугольник обозначает целевой объект, красный круг — начальную точку, а белый круг с синей стрелкой — конечную точку и его ориентацию.

Таблица 23 — Результаты отдельных тестов классического и основанного на обучении подходов на реальном роботе в сравнении с полным методом SkillFusion.

Method	Scene	Success	SPL	SoftSPL	Path Length, m	Time, s
RL	Chair 1	0.67	0.53	0.42	22	170
	Chair 2	0	0	0.42	36	300
	Sofa	1	0.39	0.39	38	190
	Average	0.71	0.39	0.41	31	200
Classic	Chair 1	0.67	0.35	0.41	28	180
	Chair 2	1	0.40	0.40	33	220
	Sofa	1	0.75	0.75	24	110
	Average	0.86	0.53	0.56	27	170
SkillFusion	Chair 1	1	0.61	0.61	25	150
	Chair 2	1	0.61	0.61	22	140
	Sofa	1	0.77	0.77	22	110
	Average	1	0.65	0.65	23	130

В данном разделе диссертационного исследования была рассмотрена проблема визуальной навигации к целевым объектам в замкнутых пространствах. Эта задача состоит в том, что воплощенный агент (виртуальный или реальный робот) должен достичь некоторого объекта, принадлежащего к предопределенному классу (например, стола, стула и т.д.) в неизвестной среде. Наличие объекта определяется локально по сенсорным датчиками агента без наличия глобальной информации о его расположении. Были рассмотрены как необучаемые (классические), так и обучаемые методы, необходимые для эффективного решения этой задачи. Проведенное экспериментальное исследование убедительно продемонстрировало, что классический и основанный на обучении подходы к навигации агента не противоречат друг другу, но взаимно дополняют друг друга. Их объединение было реализовано как в симуляторе, так и в реальных условиях при построении высокоуровневой стратегии; были выявлены сильные и слабые стороны всех используемых навыков.

С целью повышения производительности классического процесса принятия решений были настроены и улучшены различные техники исследования среды. Чтобы применить классические навыки в реальных условиях недоступности датчиков глубины и GPS, были использованы лидарные модули локализации и картирования, которые позволили достичь аналогичной производительности на реальном роботе. Чтобы повысить эффективность обучаемых навыков, была внедрена комбинированная схема обучения и новое действие «не уверен», которое не только увеличивает общие метрики, но и делает модель более устойчивой к внешним шумам. Эти улучшения также сделали обучаемые навыки переносимыми в реальную среду без какой-либо адаптации на заранее снятых сценах.

Кроме того, была продемонстрирована работа высокоуровневой стратегии на базе классических навыков и навыков, основанных на обучении, которая, опираясь на функцию полезности каждого навыка, переключается между этими навыками в процессе принятия решения в рамках одного эпизода. Предлагаемый в данном разделе механизм объединения навыков основан на фундаментальных различиях между классическими и обучаемыми процессами принятия решений. В проведенных экспериментах обучаемый подход показывает высокие результаты при эффективном исследовании больших площадей и способен извлекать больше информации из сенсоров для более эффективного исследования среды. Впрочем, он не настолько точен при поиске неисследованных областей, и его способности к запоминанию уже принятых решений сильно отстают от классического метода. Таким образом, каждый раз, когда требуется решить некоторую навигационную подзадачу наиболее эффективным способом, агент выбирает необходимый навык в соответствии с высокоуровневой стратегией. При развитии предложенного в данном разделе метода целесообразно улучшить методы реконструкцию семантической карты с применением интерактивной сегментации и планированием за пределами текущей карты с помощью частичной 3D-реконструкции неисследованной области данной сцены.

5.3 Программная реализация архитектуры NSLP для беспилотного автомобиля

Предложенная архитектура NSLP-агента позволяет реализовать адаптивную систему управления беспилотным автомобилем. На рисунке 5.20 представлена схема интеграции NSLP-агента в широко распространенную в индустрии систему управления беспилотными автомобилями Apollo. Наряду с рядом подсистем, который используется в Apollo и в STRL (локализации, построение карты и т.д.), за адаптивное планирование поведения здесь отвечает NSLP-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.



Рисунок 5.20 — Схема использование NSLP агента в качестве модуля в системе управления Apollo.

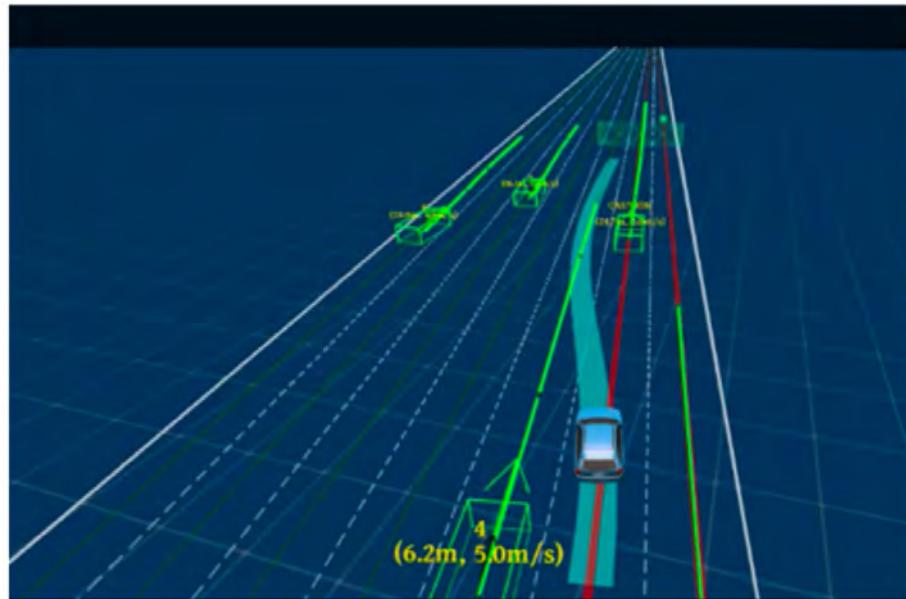


Рисунок 5.21 — Пример реализации сценария обгона динамических препятствий в системе управления Apollo с использованием предложенной концепции адаптивного планирования. Зеленые тонкие линии — предсказываемые траектории объектов, голубая полоса — планируемая траектория агента.

В данном разделе диссертационного исследования рассматриваются задачи парковки и обгона беспилотными транспортным средством (автомобилем) динамических препятствий (других автомобилей) на многополосном шоссе (см. рисунок 5.21). В данных задачах предполагается, что имеющийся 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 объектах 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 действий, которые должен совершить агент (перестроение направо, перестроение налево). Данный план строится путем поиска кратчайшего на графе поведенческого дерева (вычислительный аналог дерева Монте Карло в Apollo), в котором реализованы основные правила и условия обгона.

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

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

5.3.1 Реализация парковочного маневра

За последнее десятилетие во многих исследованиях использовались алгоритмы, предназначенные для планирования движения беспилотных транспортных средств (БТС) в различных задачах. Автоматическая парковка — это одна из задач, которая обычно описывается как движение вперед с достижением правильной ориентации транспортного средства, а затем движение назад для достижения места парковки. При разработке и тестировании алгоритмов планирования различных маневров для БТС, в том числе парковки, используются различные программные платформы, среди которых хорошо известная платформа Baidu Apollo⁸. Перед развертыванием алгоритма на реальном средстве с использованием этой платформы его часто тестируют (и при необходимости обучаются) в высокоточном симуляторе, таком как SVL [405]. Большинство ошибок реализации траектории маневра вызваны тем, что планировщик строит не реализуемую низкоуровневой подсистемой управления последовательность подцелей.

Для планирования различных маневров в Apollo используется алгоритм гибридного A*. Один из основных его недостатков заключается в том, что он генерирует небольшие фрагменты траектории с различными типами направления движения (задним и передним ходом), что приводит к большой накопленной ошибке при реализации спланированной траектории и, как следствие, к тому, что БТС может не попасть на заданное парковочное место. Вторая проблема, связанная с генерированной гибридным A* траекторией, заключается в том, что в ней не учитываются динамические препятствия, возникающие при реализации маневра парковки, что может привести к столкновению или построению неполной траектории. В данном разделе предлагается адаптировать подход одновременного планирования и обучения к решению этих проблем.

Многие современные алгоритмы планирования с обучаемыми компонентами [49; 536; 538] демонстрируют возможность генерации эффективного поведения автономной робототехнической платформы в динамичных средах. В результате анализа качества недавно разработанных алгоритмов по реализации одновременного обучения и планирования был

⁸<https://github.com/ApolloAuto/apollo>

выбран метод POLAMP [22]. Поскольку задача адаптивной парковки разделена на две подзадачи: движение вперед и движение назад, — то для эффективной реализации процесса обучения и процесса принятия решений на базе метода POLAMP требуется алгоритм генерации конечных позиций для первой подзадачи. Эта проблема решается глобальным планировщиком POLAMP.

После реализации предложенного подхода в задаче парковки он был интегрирован в платформу Apollo, которая взаимодействует с симулятором SVL с помощью ROS (CyberRT) узлов и системы сообщений. Были разработаны соответствующие CyberRT узлы, позволяющие реализовать процесс построения траектории с помощью метода POLAMP и считывать необходимую информацию из других подсистем Apollo и соответствующих им узлов. Предложенный алгоритм был протестирован на нескольких стандартных картах с различными сценариями парковки, такими как парковка на автостоянке. Полученные результаты показывают, что в ситуациях с динамическими препятствиями предложенный подход превосходит стандартный планировочный алгоритм.

Планирование маневра с помощью гибридного алгоритма A* и обучаемых методов

Недавно предложенные методы по генерации траектории маневра, такие как [490], представляют собой подходы, предполагающие генерацию траектории определенного вида и использующие фиксированную форму участков траектории для сокращения времени выполнения задачи генерации маневра. Такие методы также не учитывают возникновение динамических препятствий, хотя они очень эффективны при отсутствии препятствий с точки зрения затрачиваемого на генерацию траектории времени.

Другим подходом к планированию маневров являются гибридный алгоритм A* и его варианты реализации [558]. Гибридный A* является встроенным планировщиком в платформе Apollo. Он основан на алгоритме A* и реализует процедуру поиска целевого состояния в клеточном пространстве. Основное различие между A* и гибридным A* заключается в том, что узлы, которые выбираются из списка Open на каждой итерации алгоритма, имеют ограничения, накладываемые простой велосипедной моделью кинематики. Примитив движения задается как параметр, определяющий, на какое расстояние агент может переместиться из текущего состояния за время $t = 0,1\text{c}$, с десятью промежуточными точками (данний параметр может быть изменен в конфигурациях). Эти десять точек выбираются при генерации движения вперед, другие десять точек выбираются при генерации движения назад. В дополнение к этому изменению встроенный планировщик при переходе к следующей точке использует систему штрафов на основе следующей системы правил: штраф за переключение передач l_1 ; штраф за искривление траектории l_2 ; штраф за изменение угла поворота l_3 ; штраф за движение вперед l_4 ; и штраф за движение задним ходом l_5 .

Поскольку при рассмотрение узлов из списка Open не учитывается движение других автомобилей, ни гибридная версия A*, ни собственно сам алгоритм A* не могут работать с динамическими сценариями. Однако динамические препятствия часто встречаются в сценариях парковки, что делает алгоритмы на основе обучения с подкреплением более предпочтительными по сравнению с классическими методами.

В работе [389] было предложено решить проблему навигации из точки A в точку B при наличии динамических препятствий с помощью функции вознаграждения:

$$R_{\theta_r} = \theta_r^T [r_{step} r_{goalDist} r_{collision} r_{turning} r_{clearance} r_{goal}],$$

которая отвечает за то, чтобы агент избегал препятствий, был оштрафован за столкновение и вознагражден за приближение к цели.

Использование архитектуры обучения «актор-критик» позволяет агенту изменять параметры стратегии планировщика на каждом шаге обучения, а не только в конце эпизода, как в случае с обычными градиентными методами. Задача критика состоит в том, чтобы итеративно обновлять функцию полезности действия Q , оценивая действия, генерируемые актором, на каждой итерации.

Метод, описанный в работе [538], следует выделить как один из более продвинутых современных подходов по обучаемой генерации маневров. Авторы предложили использовать следующую функцию вознаграждения:

$$R = R_{goal} + R_{timestep} + R_{collision} + R_{potential} + R_{waypoint},$$

которая принципиально отличается от предыдущего метода. Кроме того, авторы предлагают использовать планировщик, который делит глобальную траекторию от начального до конечного состояний на промежуточные подцели. Вознаграждение устанавливается таким образом, чтобы агент эффективно достигал следующей подцели, одновременно избегая динамические препятствия. Это делается для того, чтобы у локального планировщика был некоторый допустимый диапазон для модификации примитивов перемещений. Было продемонстрировано, что при такой конфигурации метода, обученного на упрощенной двумерной среде, стратегия хорошо переносится в ранее не наблюдавшую реалистичную трехмерную среду.

Также необходимо отметить ряд работ, в которых обучение с подкреплением используется для построения траектории маневра [15; 30; 39] или для навигации мобильного робота [11; 74]. Однако в этих работах не используется интеграция с известными платформами для беспилотных транспортных средств.

Применение алгоритма POLAMP для задачи парковки

В данном разделе диссертационного исследования используется алгоритм POLAMP [22] в качестве локального планировщика, который был обучен в сценариях по избеганию

динамических препятствий. Сам метод включает в себя реализацию алгоритма РРО [512] с аппроксимацией состояния в виде последовательности наблюдений (frame stack). Велосипедная модель была интегрирована в симулятор, где проводится обучение метода POLAMP, а ограничения на возможные ускорения были использованы для того, чтобы сделать движение агента более реалистичным и приближенным к динамике БТС в системе Apollo.

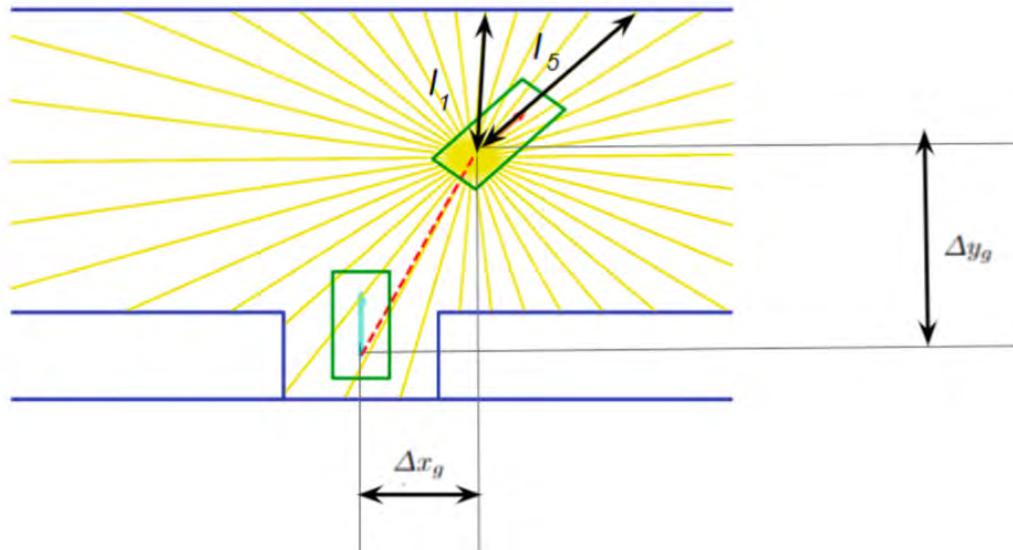


Рисунок 5.22 — Элементы вектора наблюдения для локальной обучаемой стратегии.

Как обычно, предполагается, что стратегия π генерирует действия по наблюдениям в каждый момент времени $a_1 = \pi(o_0), \dots, g = \pi(o_{n-1})$. Предполагается, что стратегия параметризуется с помощью набора параметров θ . Алгоритм градиента стратегии реализует процесс оптимизации параметров θ , чтобы максимизировать ожидаемую будущую отдачу и минимизировать длину траектории:

$$\pi^* : \begin{cases} \pi^* = \arg \max_{\pi(\theta)} \mathbb{E} [\sum_0^\infty \gamma^t r(s_t) | \pi], \\ \pi^* = \arg \min_{\pi(\theta)} \|\{\pi(o_0), \pi(o_1), \dots, \pi(o_{n-1})\}\|, \\ g = \pi(o_{n-1}). \end{cases} \quad (5.6)$$

Поскольку окружающая агента среда является частично наблюдаемой в каждый момент времени t , действие a_t генерируется из распределения $\pi^*(o_t)$ по наблюдению o_t . Наблюдение представляет собой вектор $o_t = (\Delta x_g, \Delta y_g, \Delta\theta, \Delta v, \Delta\gamma, \theta, v, \gamma, flag, l_1, l_2, l_3, \dots, l_n)$, где $\Delta\theta$ — угол направления движения, Δv — разница между скоростью агента и ограничением скорости на парковочном месте, $\Delta\gamma$ — разница между углом поворота агента и ограничением угла поворота на парковочном месте, $flag$ — информация о типе выполняемой задачи (задача движения вперед, задача движения задним ходом), добавленная для упрощения обучения, $l_1, l_2, l_3, l_4, \dots, l_n$ — это симуляция лидарных данных, состоящая из n лучей, и координаты цели $\Delta x_g, \Delta y_g$, как показана на рисунке 5.22.

Для того чтобы алгоритм обучения сгенерировал соответствующий профиль скорости и ускорения для траектории, необходимо представить действия агента в виде пары (a, ε) , где

a — линейное ускорение, а ε — угловое ускорение. Функция вознаграждения была составлена следующим образом:

$$R = w^T [r_{collision}, r_{timestep}, r_{distance}, r_{overspeeding}, r_{oversteering}, r_{action_a}, r_{action_\varepsilon}],$$

где $r_{collision}$ равно -20 , когда агент сталкивается с препятствиями, и 0 в противном случае, $r_{timestep}$ — постоянный штраф за каждый шаг со значением 1 , $r_{distance}$ — евклидово расстояние до цели, $r_{overspeeding}$, $r_{oversteering}$ равны -5 , когда агент пытается превысить ограничение скорости, а $r_{action_a}, r_{action_\varepsilon}$ являются штрафами за абсолютное значение линейного и углового ускорения, которые равны $-0,5 * a_t - 0,5 * \varepsilon_t$ в моменты времени t .

В качестве локального планировщика метода POLAMP был использован алгоритм РРО. В данном алгоритме актор включает в себя нейросетевой аппроксиматор, принимающий на входе наблюдение o_t и возвращающий математическое ожидание и дисперсию линейного ускорения и угловой скорости. Во время обучения актор обновляет веса стратегии π в соответствии с градиентом функции полезности стратегии:

$$\Delta_w J(w) = E_{\pi_w} \Delta_w \log \pi_w(s, a) A^{\pi_w}(s, a).$$

Критик используется для прогнозирования функции преимущества $A^\pi(o_t, a_t)$, которая указывает, насколько предпочтительнее выбрать действие a_t при наблюдении o_t . С учетом наличия критика актор обновляет свои веса в соответствии с усеченной функцией потерь:

$$L(s, a, w_k, w) = \min(\pi_w(a|s)/\pi_{wk}(a|s) \cdot A^{\pi_{w,old}}(s, a), \text{clip}(\pi_w(a|s)/\pi_{w,old}, 1 - \varepsilon, 1 + \varepsilon) A^{\pi_{w,old}}(s, a)).$$

Веса критика обновляются путем оценки $A^\pi(s_t, a_t)$ по сгенерированным траекториям.

Динамическая модель агента была задана с помощью велосипедной модели с референсной точкой в центре задней оси модели транспортного средства:

$$\begin{aligned} \dot{x}_r &= v * \cos(\theta), \\ \dot{y}_r &= v * \sin(\theta), \\ \dot{\theta} &= \frac{V * \tan(\delta)}{L}. \end{aligned} \tag{5.7}$$

Кинематические ограничения были заданы следующим образом:

$$\begin{aligned} -2 \text{ m/s} &\leq v \leq 2 \text{ m/s}, \\ -1.5 \text{ m/s}^2 &\leq a \leq 1.5 \text{ m/s}^2, \\ -1.5 \text{ rad/s}^2 &\leq \varepsilon \leq 1.5 \text{ rad/s}^2. \end{aligned} \tag{5.8}$$

Ограничения для оценки достижимости парковочного места (терминальные ограничения) были следующими:

$$\begin{aligned} distance &\leq 0.5 \text{ m}, \\ -0.5 \text{ m/s} &\leq v \leq 0.5 \text{ m/s}. \end{aligned} \tag{5.9}$$

Для организации процесса обучения был использован подход, основанный на заранее составленной программе обучения, чтобы сформировать эффективную стратегию агента по

парковке в сценариях со статическими и динамическими препятствиями. Для обучения были выбраны два основных этапа (см. рисунок 5.23): обучение в пустой среде со статичными препятствиями и совместное обучение, когда агенту назначается место парковки в качестве цели после того, как агент достигнет первую подцель. На каждом этапе также есть несколько последовательных этапов обучения. Например, чтобы агент научился достигать цели с ограничениями по расстоянию, углу направления и скорости, его сначала обучали только с ограничениями по расстоянию.

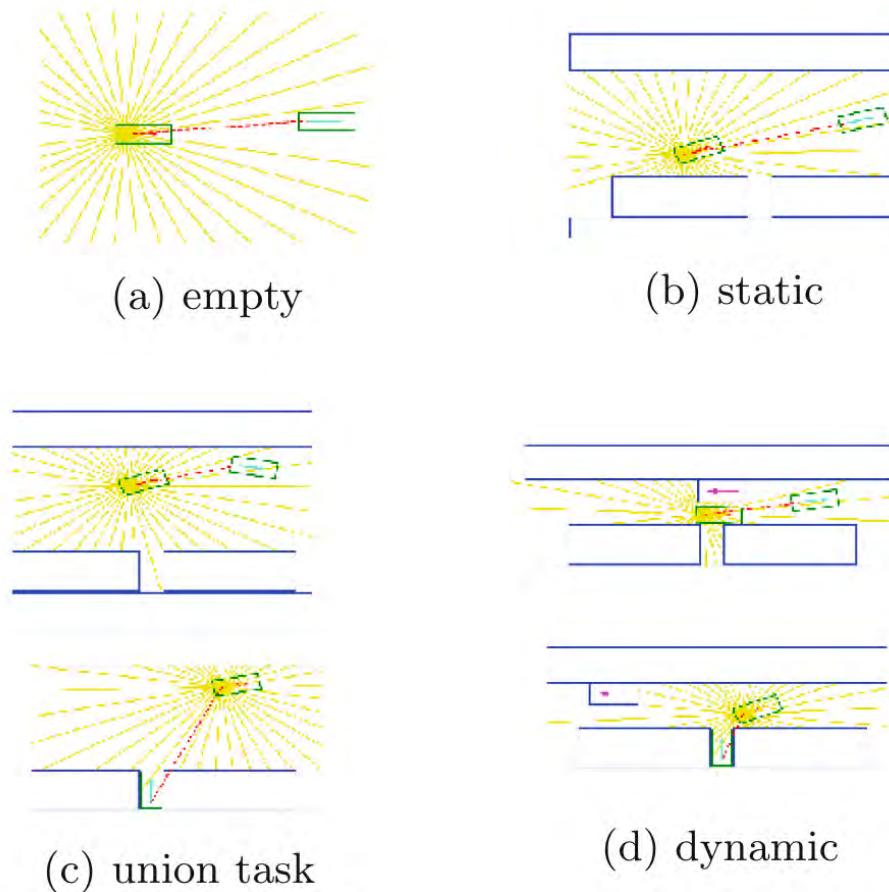


Рисунок 5.23 – Этапы программы обучения стратегии агента.

Обучение стратегии агента проходило в четыре этапа, однако количество итераций обучения на каждом этапе может варьироваться. Например, чтобы получить стратегию агента по достижению места парковки со всеми ограничениями, агент был сначала обучен в пустой среде только с одним условием $distance \leq 0,5$ м, затем было добавлено условие на угол $angle \leq 15^\circ$ и условие на скорость $speed \leq 0,5$ м/с.

На рисунке 5.24 изображена архитектура нейросетевого аппроксиматора, используемая в процессе обучения. Три последовательных наблюдения агента o_{t-1} , o_{t-2} и o_{t-3} объединяются с текущим наблюдением o_t . Поскольку используется $n = 39$ лидарных лучей, вектор наблюдения имеет длину 48. Когда происходит объединение предыдущих наблюдений, получается вектор длиной $48 * 4 = 192$. При обучении актера на нейросетевой аппроксиматор поступает пакет данных в размере 8000. При обучении критика действие агента было добавлено к входному вектору, так что его размер равен 194.

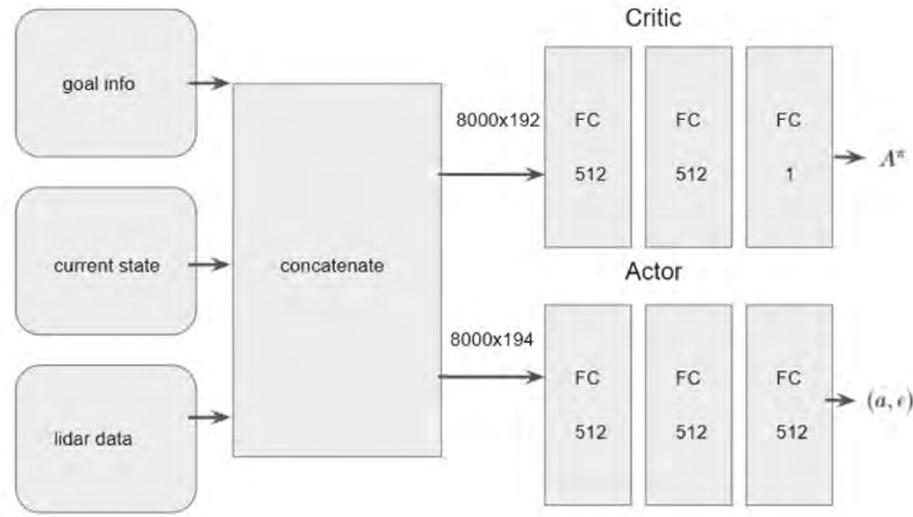


Рисунок 5.24 — Архитектура нейросетевых аппроксиматоров актора и критика.

5.3.2 Экспериментальное исследование в составе платформы Apollo

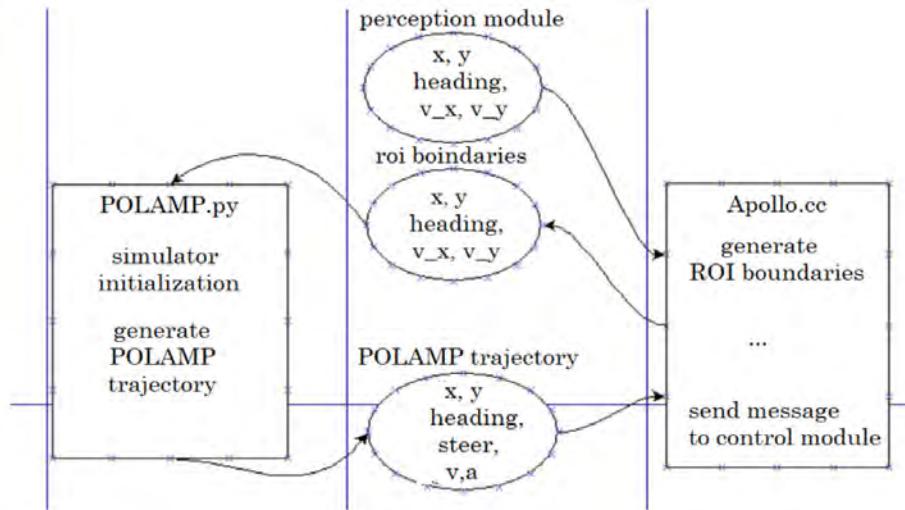


Рисунок 5.25 — Модули восприятия, определения границы сцены, генерации траектории POLAMP в виде топиков CyberRT.

Алгоритм генерации действий POLAMP был встроен в платформу Apollo путем организации взаимодействия с имеющимися топиками CyberRT на чтение и запись (см. рисунок 5.25). Для инициализации внутреннего симулятора POLAMP требуется информация о статических препятствиях в среде и о границах сцены, где может перемещаться агент. Эта информация предоставляется в Apollo с помощью алгоритма ROI (детекция области интереса). Информация о динамических препятствиях поступает в алгоритм принятия решений ROI через модуль восприятия, реализованный в Apollo, который получает и публикует каждые 0,1 секунды положение динамических препятствий и векторы их скоростей. Когда данная информация поступает в симулятор планировщика POLAMP,

запускается процесс обучения POLAMP. После этого агент генерирует один эпизод, исходя из предположения о равномерном прямолинейном движении динамических препятствий, сохраняет точки траектории в виде кортежей (x , y , курсовой угол, угол поворота руля, линейное ускорение, угловое ускорение) и предоставляет полученную траекторию в модули Apollo, реализующие движение по этой траектории.

Профиля скорости и ускорения траектории сглаживаются стандартными алгоритмами Apollo, поскольку скорость в конечной точке каждой траектории POLAMP для движения вперед и назад может варьироваться от $-0,5$ до $0,5$ м/с. Если конечная точка текущей траектории имеет индекс i и если конечная точка парковочного места имеет индекс j , то новое линейное ускорение и скорость для этой точки вычисляются следующим образом:

$$\begin{aligned} a(i-1) &= -v(i)/dt, \\ v(i) &= 0, \\ a(i) &= v(i+1)/dt, \\ a(j-1) &= -v(j-1)/dt, \\ v(j) &= 0, \\ a(j) &= 0. \end{aligned} \tag{5.10}$$

На рисунке 5.26 показаны результаты, которые были получены после обучения алгоритма в симуляторе POLAMP.

Были оценены следующие основные показатели: среднее вознаграждение за эпизод (среднее вознаграждение по всему пакету обучения — 8000 заданий) и показатель успешности выполненного задания — количество выполненных заданий в наборе валидационных данных (включает 600 заданий). В случае сцен с узкими дорогами основными случаями, когда агент не завершает эпизод, являются ситуации, в которых агент должен зафиксировать направление вектора движения, чтобы добраться до места парковки со всеми ограничениями, что можно считать редким случаем для обычной парковки.

Был обучен и протестирован алгоритм генерации маневра с использованием платформы Apollo и симулятора SVL. В сценариях возможного столкновения агента с динамическим препятствием алгоритм успешно научился их избегать. Если препятствий нет, алгоритм находит оптимальную траекторию, как показано на рисунке 5.27.

Среди неуспешных случаев выполнения представленного алгоритма в основном встречались сценарии, где была необходимость в реализации движения задним ходом, что приводило к формированию траектории с некоторой ошибкой. Эта проблема возникает из-за того, что встроенный модуль управления не может отслеживать профиль скоростей и ускорений, обеспечиваемый алгоритмом POLAMP.

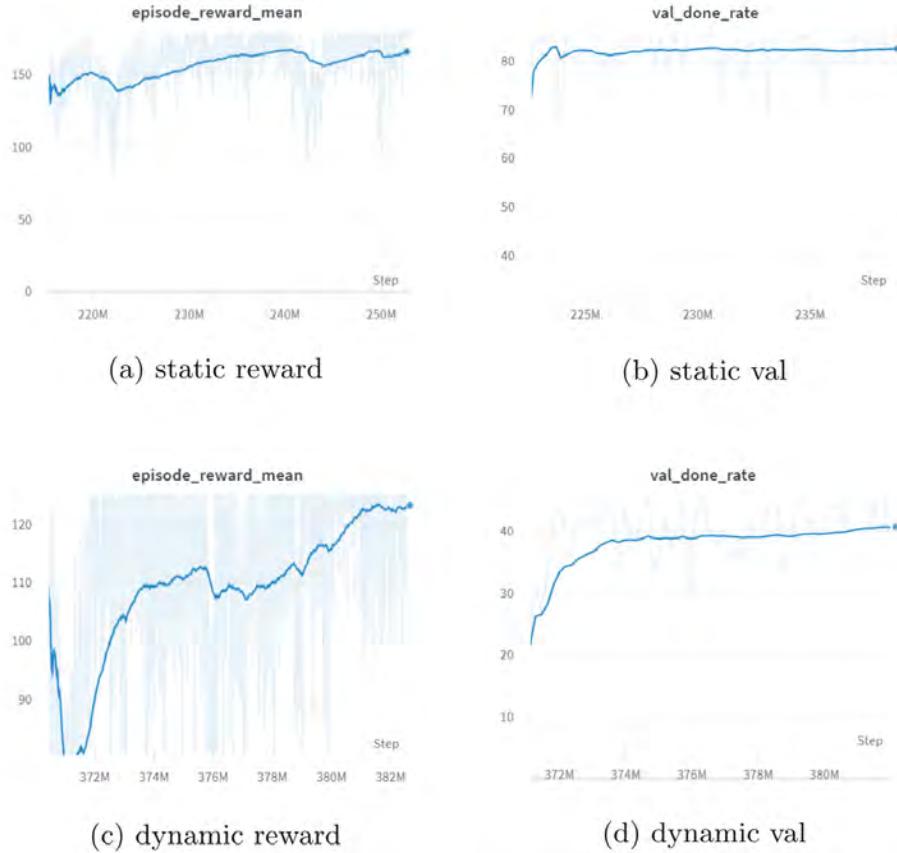


Рисунок 5.26 — Результаты обучения алгоритма POLAMP в задаче генерации маневра парковки.

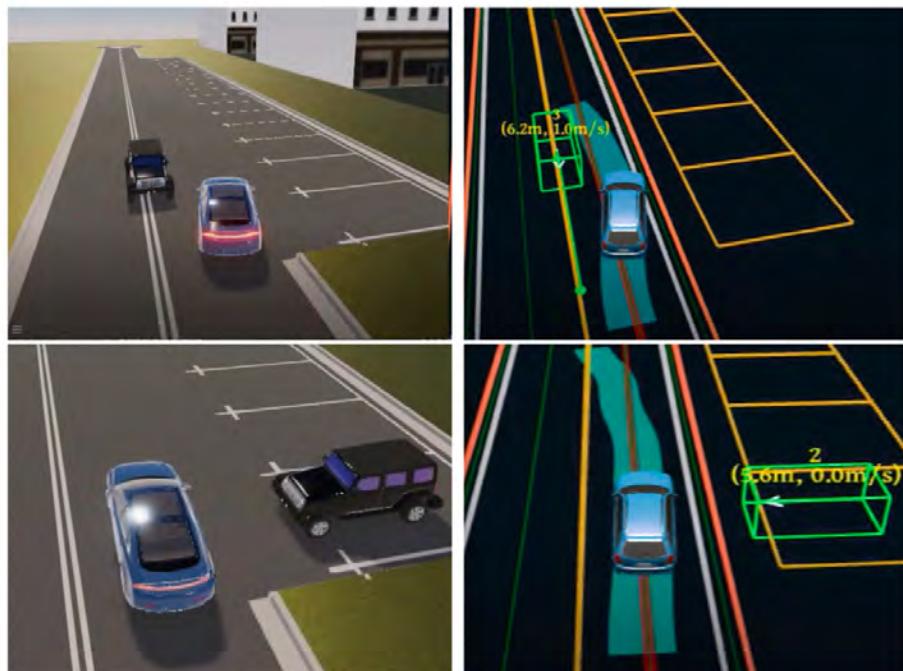


Рисунок 5.27 — Различные сценарии парковки с динамическими препятствиями были протестированы в симуляторе SVL (рисунки слева) и в Apollo (рисунки справа).

5.4 Выводы

В данном разделе диссертационного исследования были рассмотрены две реализации архитектуры NSLP в составе более общей архитектуры STRL по управлению воплощенными агентами. Первая реализация с использованием промежуточного программного обеспечения ROS была создана для управления мобильным роботом с манипулятором. Данная реализация содержит полноценные реализации реактивного и тактического уровня с лидарной одометрией и локализацией и модулем построения карты препятствий. На стратегическом уровне реализована концепция одновременного обучения и планирования в NSLP-архитектуре. Полученная общая архитектура под названием STRL-Robotics была протестирована в задаче навигации в человеко-ориентированных средах к целевым объектам, а также с необходимостью совершения дополнительных действий для манипуляции объектами (например, нажатие кнопки лифта).

Вторая реализация с использованием промежуточного программного обеспечения Apollo была создана для управления беспилотным транспортным средством в задаче генерации маневров обгона и парковки. Подробнее был рассмотрен процесс обучения именно для маневра парковки, для которого был адаптирован метод POLAMP. Получаемые адаптивные стратегии показывают более качественное поведение в сценариях с динамическими препятствиями по сравнению со стандартными планировщиками на основе гибридного алгоритма A*.

Глава 6. Когнитивные аспекты объектно-центричного представления

В данной главе представлена когнитивная составляющая предложенной архитектуры NSLP с точки зрения использования языковых моделей для составления высокоуровневого плана действий. Основные результаты, изложенные в данной главе, были опубликованы в [45; 69; 78] (раздел 6.1), [24] (раздел 6.2).

6.1 Знаковая картина мира как нейросимвольная модель

В данном разделе диссертационного исследования рассматривается один из вариантов применения знаковой модели, а точнее, теории субъективных знаковых моделей действительности, к задаче построения архитектуры агента, взаимодействующего со средой. Предложена знаковая реализация нейросимвольной архитектуры NSLP когнитивного агента, в которой центральное место уделено решению проблемы привязки символов за счет использования понятия знака в его психологической интерпретации. Предложены более строгие формулировки методов синтеза плана поведения и приобретения знаний при использовании знакового (семиотического) подхода. Дано новое, уточненное определение сценария поведения, и предложены пути его использования при построении плана поведения.

6.1.1 Семиотическая реализация нейросимвольной интеграции

В данном разделе ставится цель по разработке семиотической реализации архитектуры NSLP когнитивного агента, действующего в динамической среде, о которой агенту не предоставляется никакой априорной информации. Основной особенностью предлагаемой архитектуры является то, что выработка рационального поведения агентом происходит за счет автоматического приобретения знаний о правилах поведения в среде. Агенту заранее не выдается описание целей и правил поведения в среде. Единственным критерием оптимальности служит только специальная функция подкрепления, интерпретирующая внутренние и внешние сигналы по отношению к агенту. Формулировка цели, построение плана ее достижения являются частью алгоритма синтеза поведения агента. Этап обучения агента и накопления опыта выполнения планов действия является в данном случае ключевым. Только в результате обучения агент приобретает возможность определять цель своей деятельности в заданной среде и повышать эффективность вырабатываемых действий. Предложенная архитектура может иметь и конкретный прикладной интерес в связи с задачей имитации работы человека, взаимодействующего с некоторым сложным объектом

управления, а именно для моделирования процесса восприятия поступающей информации и поиска среди представленной информации необходимых данных для решения проблемы.

Компоненты семиотической архитектуры

При построении архитектуры агентов, взаимодействующих со средой, одним из ключевых понятий является так называемый когнитивный цикл [100; 488] — замкнутая последовательность операций, выполняемых агентом в процессе своего функционирования (см. раздел 2.1). Описание такой последовательности обычно предполагает выделение модулей, блоков сохранения и обработки информации, которые связаны друг с другом функциональными переходами, подразумевающими дополнительную обработку и обмен информацией. Во многом это связано с историческими предпосылками, когда идея об отдельных автономных функциональных областях мозга, отвечающих за ту или иную когнитивную функцию, переносилась и на структурную организацию архитектуры агента. Такой функциональный модульный подход во многом является основным препятствием на пути построения интегрированной непрерывной нейросимвольной модели. В отличие от этого, в данном разделе предлагается сконцентрироваться на процессе взаимодействия со средой, который во многом определяет строение всей архитектуры когнитивного агента и позволяет определить основные структурные составляющие элемента индивидуального знания агента.

Далее представлено более подробно описание архитектуры семиотического агента, изображенной на рисунке 6.1. При построении модели, пригодной для реализации в робототехнических задачах, необходимо учитывать то, что когнитивный (семиотический) агент является воплощенным, т.е. он взаимодействует с предметной внешней средой через посредство внутренней среды, граница между которыми может быть проведена достаточно условно. Агент обрабатывает информацию, используя сигналы, поступающие из внутренней среды. Часть сигналов внутренней среды на самом деле вызывается сигналами, поступающими из внешней среды. Предполагается, что в результате инициализирующего обучения агенту доступен механизм разделения поступающих признаков на внутренние и внешние. Агент влияет на внешнюю среду, меняя состояние внутренней среды, выполняя моторные действия, меняющие характер протекающих во внешней среде процессов. Агент имеет доступ к языковой системе (опосредованной, например, языковой моделью — см. разделы 2.4 и 6.2), являющейся источником имен представляемых агентом сущностей внешней и внутренней среды и правил построения сообщений, посредством которых поддерживается коммуникация с другими участниками деятельности. Далее неявно будет использоваться соглашение о том, что каждой сущности, которую агент может представить на концептуальном уровне, однозначно сопоставляется некоторое имя. Поскольку в данном разделе целью является моделирование рационального поведения агента, то естественно

предположение, что во внутренней среде и на концептуальном уровне представлений агента существуют цели, на достижение которых направлена деятельность агента.

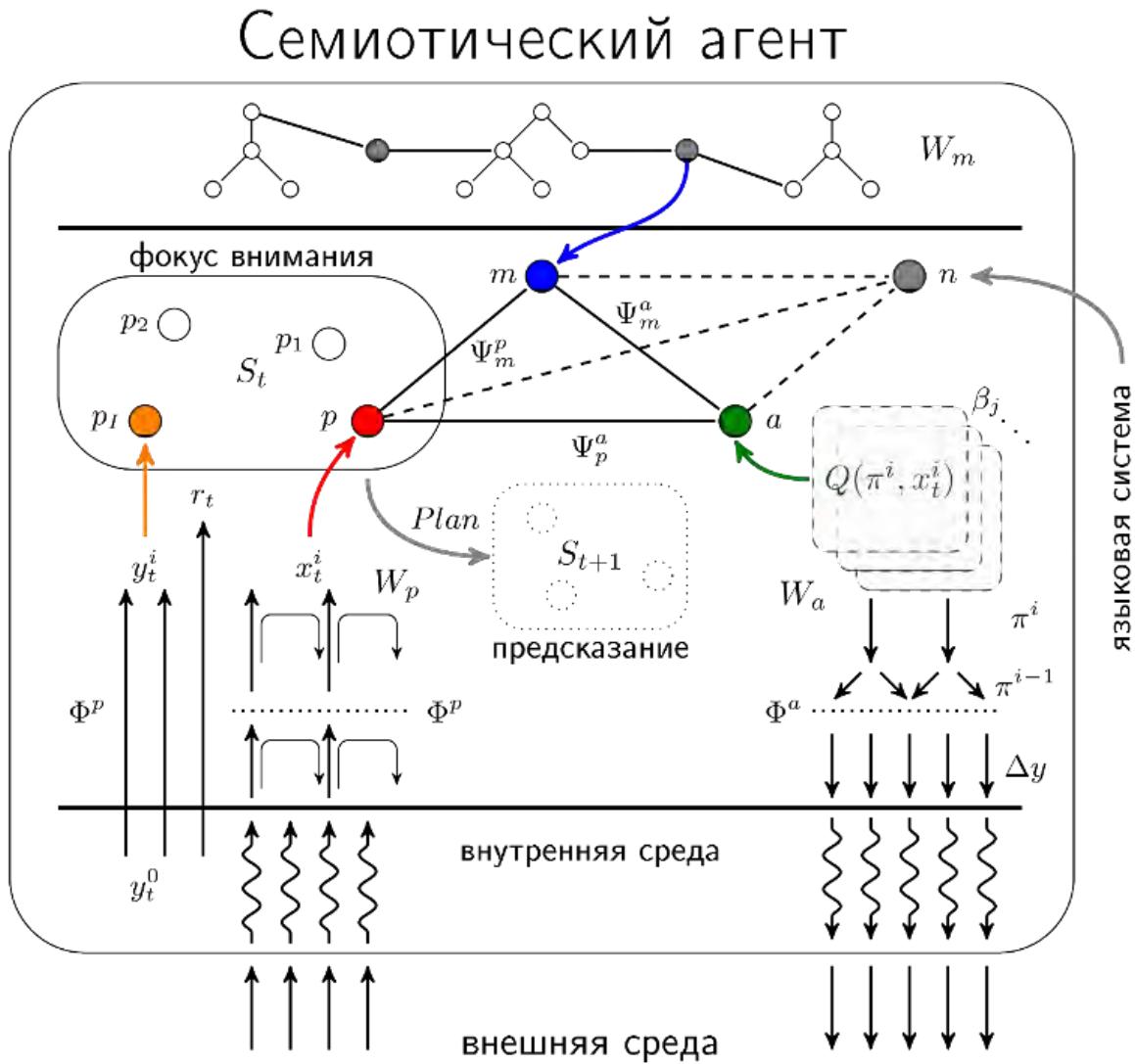


Рисунок 6.1 — Концептуальная схема работы когнитивного семиотического агента.
Обозначения расшифровываются в основном тексте.

Несмотря на то что в действительности поток поступающих сигналов на сенсоры агента, а также воспроизводимые агентом моторные действия являются непрерывными, будет использоваться временная дискретизация входных данных и их типизация в зависимости от принимающих их сенсоров и генерирующих актуаторов. Получаемые таким образом порции данных будут называться элементарными признаками и их значениями, которые интерпретируются агентом как признаки сущностей внешней среды (внешние признаки) или как признаки внутренней среды (внутренние признаки или признаки «Я»).

Вектором x_t^0 размерности l_{in}^0 (определяется строением сенсоров агента) будет обозначаться набор значений элементарных внешних признаков в момент времени t , вектором y_t^0 размерности l_{out}^0 (определяется строением внутренней среды агента) — набор значений элементарных внутренних признаков в момент времени t . Под значением будет пониматься уверенность в детектировании данного признака (от 0 до 1). Агент

реализует функцию категоризации поступающего набора данных за промежуток времени T (условно с момента времени $t = 0$), соотнося последовательности векторов $x_{0:T}^0$ и $y_{0:T}^0$ со своим набором признаков тех сущностей, которые он ранее наблюдал. Результатом работы такой процедуры распознавания Φ^p является подмножество высокоуровневых признаков $\Phi^p(x_{0:T}^0, y_{0:T}^0) = p_i(n) \subset P$, которые будут называться образами сущностей, представляемых агентом и соотнесенных с именами n . Множество P — это множество всех образов, которые может распознавать агент. В семиотическом агенте вычисление процедуры Φ^p организовано на основе *каузальной предикторной сети* иерархическим образом с использованием биологически правдоподобных механизмов распространения активности (вывода и обучения), предсказания и подавления второстепенного сигнала (см. подробнее далее).

Сущности с именами n_i , которые соотнесены с помощью процедуры распознавания Φ^p с текущими входными данными, образуют множество активных сущностей S_T в момент времени T . Далее будут представлены и другие механизмы пополнения множества S_T . Среди сущностей S_T присутствует выделенная сущность с именем «Я» (I), которая является результатом категоризации только внутренних признаков $\Phi^p(y_t^0) = p_I = p(I)$.

Элементарные моторные действия (команды) агента будут описываться как фиксированное «мгновенное» изменение определенного набора элементарных внутренних признаков агента $\Delta y_t = y_{t+1}^0 - y_t^0$. Здесь вектор y_{t+1}^0 задает акцептор результата действия, а y_t^0 — его условия. Набор конкретных моторных команд в промежуток времени с $t = 0$ до $t = T$ определяется с помощью иерархической процедуры разворачивания действия $\Phi^a(a(n)) = \Delta y_{0:T}^0$, где $\Delta y_{0:T}^0 = \pi^0$ — это стратегия агента на уровне моторных действий, т.е. последовательность команд $\Delta y_0, \Delta y_1, \dots$, а $a(n) \in \pi^{j+1}$ — это часть стратегии агента верхнего уровня, т.е. высокоуровневое действие. Действие $a(n)$ уже не является «мгновенным» и реализуется за время T за счет иерархии операций, на нижнем уровне которой находится последовательность команд $\Delta y_{0:T}^0$. Так как рассматриваются условия предметной целенаправленной деятельности, в предлагаемой модели каждое высокоуровневое действие $a(n)$ ассоциировано с сущностью с именем n . Под ассоциацией понимается то, что представляемая агентом сущность с именем n определяет условия выполнимости действия $a(n)$. Определение данных условий в модели семиотического агента реализуется с помощью функции связывания $\Psi_p^a(p(n)) = a(n)$, которая связывает образ предмета с действием $a(n)$, иными словами, по текущим активным признакам (признакам, которые в данный момент наблюдаются в потоке данных) определяется актуальная реализация действия, т.е. низкоуровневая стратегия $\pi^j = a_{0:T}^j$.

Любое действие кроме условий выполнения (запуска) предполагает задание результата [675], который определяет, успешно ли завершилось выполнение действия. При моделировании рационального поведения такое условие задается акцептором результата (в стиле функциональных систем Анохина) или целью действия. С учетом сказанного считается, что в модели семиотического агента любое предметное действие является частью смысла $a(n)$ сущности, представляющей агентом и соотнесенной с именем n , и определяется парой π, β , где π — стратегия или операционный состав действия (последовательность

действий более низкого уровня иерархии), β — цель выполнения действия (акцептор результата). Множество смыслов A всех представляемых агентом сущностей в семиотическом агенте реализуется с помощью *каузальной акторной сети*, обновление которой происходит с помощью иерархического обучения с подкреплением (см. подробнее далее).

Взаимодействие смысла и образа сущности с именем n происходит не только посредством функции связывания Ψ_p^a . Оказывается возможным задать обратную функцию $\Psi_a^p = (a(n)) = p(n')$, которая позволяет поставить в соответствие акцептору результата действия $a(n)$ набор признаков, сопоставляемых с образом $p(n')$ предмета-цели или предмета потребности с именем n' .

При построении архитектур агентов, которые обучаются с подкреплением, оказывается, что такие автоматические переходы Ψ_p^a и Ψ_a^p эффективно работают только в достаточно простых условиях, которые не предполагают глубокой иерархии образов и смыслов [512; 580]. Для синтеза более сложного поведения в когнитивных архитектурах оказывается необходимым формировать отдельную память, в которой моделируются особенности функционирования внешней среды. Модули памяти, в которой хранятся значимые причинно-следственные отношения, выявляемые во внешней среде, известны во многих когнитивных и общих архитектурах агентов [175; 371]. Методы обучения с подкреплением, основанные на модели, также задействуют различные варианты представления модели реакций внешней среды [189; 382]. В предлагаемой семиотической реализации модуля одновременного планирования и обучения архитектуры NSLP для представления агентом данной информации используются сценарии [435; 496] взаимодействия агента со средой и другими участниками деятельности, которые организованы в *сценарную каузальную сеть* (см. подробнее далее).

Сценарий состоит из нескольких шагов, или этапов, которые связаны причинно-следственными переходами. Ядром одного из этапов сценария $m(n)$ является обобщенная ролевая схема действия или значение сущности n , что соответствует изначальному положению о предметности деятельности агента. Значение сущности с именем n связано с ее образом и смыслом с помощью функций связывания Ψ_m^a и Ψ_m^p (и им обратным). Функция $\Psi_m^a(m(n)) = a(n)$ производит конкретизацию обобщенной схемы действия, в результате чего возникает конкретное действие $a(n)$, потенциально выполнимое в текущих условиях. Обратная функция $\Psi_a^m(a(n)) = m(n)$ позволяет сопоставить текущую реализацию действия с его ролевой схемой. Функция $\Psi_p^m(p(n)) = m(n)$ ставит в соответствие признакам образа сущности ее значение, т.е. некоторый этап сценария, в котором участвует эта сущность. Наконец, обратная функция $\Psi_m^p(m(n)) = p(n)$ по схеме действия одного из этапов сценария позволяет получить один из вариантов признакового состава сущности с именем n . В отличие от образа и смысла значение непосредственно не связывается с сенсорами и актуаторами агента, т.е. значение не является «заземленной» (привязанной) частью информации о сущности.

Введенные функции связывания $\Psi_p^m, \Psi_m^a, \Psi_a^p$ и др. (см. рисунок 6.2) параметризованы в первую очередь именем сущности n , т.е. эти функции выдают свой результат в зависимости от того, в рамках представления о какой сущности эти отображения рассматриваются в

данный момент. Во-вторых, функции связывания зависят от текущего контекста, т.е. состава фокуса внимания агента S_T (множества активных сущностей) в данный момент времени T . Эта зависимость будет записываться как $\Psi_p^m(n, S_T)$.

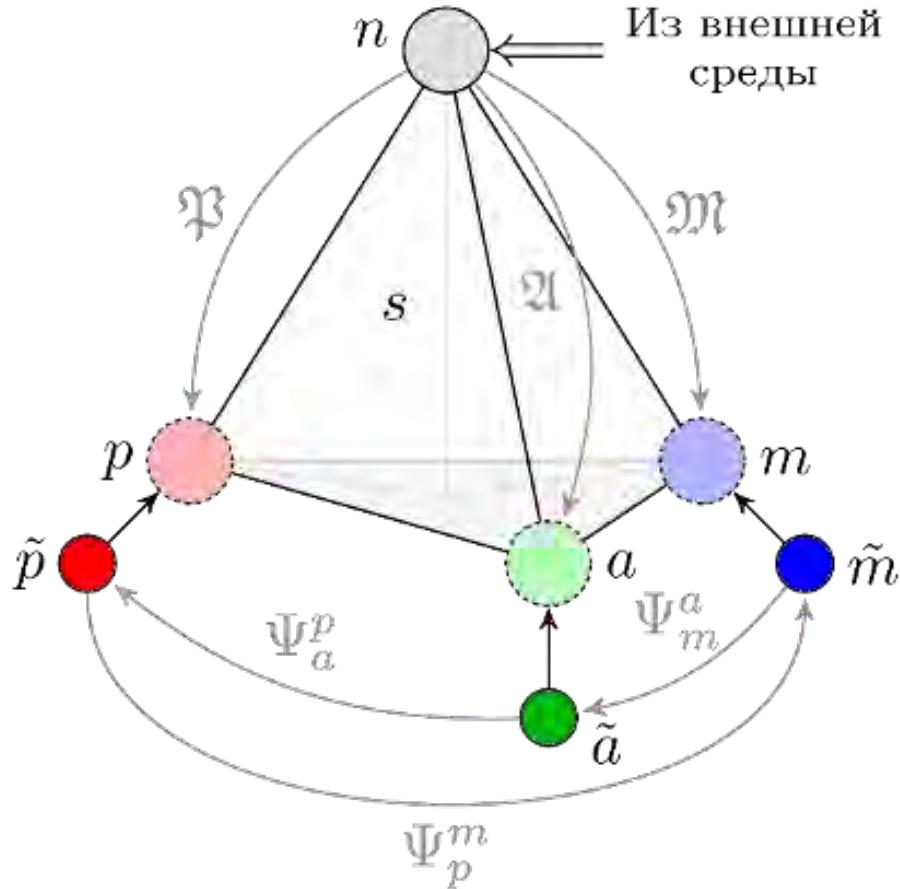


Рисунок 6.2 — Схема функций связывания Ψ_p^m , Ψ_m^a , Ψ_a^p . С тильдой обозначены узлы каузальной сети до связывания в знак. Функции именования компонент: Ψ (для образа), M (для значения), A (для смысла).

Введенные выше составляющие общей информации о некоторой сущности с именем n : образ, смысл и значение — благодаря функциям связывания образуют единую структуру, которая называется знаком $s(n)$, представляющим данную сущность. Таким образом, знак является динамической структурой, состав которой определяется благодаря текущему контексту и текущих значений функций связывания. Знак с именем n в момент времени T , в который получены сигналы $x_{0:T}^0$ и выполнены команды $\Delta y_{0:T}^0$, представляет собой четверку $n, pn, mn, a(n)$, такую, что $\Psi_p^m(n, S_T) = \Psi_a^m(n, S_T) = m(n)$ и $\Phi^a(a(n)) = \Delta y_{0:T}^0, p(n) \in \Phi^p(x_{0:T}^0, y_{0:T}^0)$.

Психологическая интерпретация

Практически все существующие подходы по решению проблемы привязки символов исходят из того, что символ как таковой является некоей неделимой сущностью, которую предлагается искусственно привязать к некоторому действительному вектору, над множеством которых вводится семейство операций [178; 234]. Эти операции могут быть интерпретированы как логические операции над множеством символов. Предполагается, что такой вектор может быть получен в результате обучения некоторой модели на основе примеров, описываемых низкоуровневыми признаками. Основная проблема, которую пытаются разрешить в подобного рода работах, — согласование полученного в ходе обучения, с видом преобразований, используемых при моделировании символьных операций. При этом такой подход по сути является декларативным, и связь между символом (именем) и задающим семантику вектором остается произвольной.

При построении архитектуры семиотического агента предлагается уйти от ограничений целостной природы символа и постулируется, что у элементов индивидуального знания агента, называемых знаками, есть структура, и их система обладает активностью, необходимой для реализации основных когнитивных функций агента. Приведенная выше архитектура семиотического агента соответствует психологической интерпретации понятия знака в духе теории деятельности [673] и культурно-исторического подхода [662]. В этом контексте деятельность является предметной и иерархически организованной. Знак представляет причинно-следственную информацию, связанную с конкретным предметом внешней среды.

Отталкиваясь от психологической интерпретации структурированного знака [686; 688], можно определить следующие основные характеристики семиотического агента:

1. Элементом концептуального уровня представления субъективных знаний агента является знак, заданный именем, определяющим его уникальность, и тремя компонентами: образом, значением и смыслом. Участие всех четырех компонентов знака в построении четырех типов каузальных сетей определяет связность и активность системы знаков (см. рисунок 6.2).
2. Образ знака определяет его связь с внешними для агента сигналами, поступающими из внешней среды. Образ, с одной стороны, определяет иерархическую признаковую структуру знака, которая строится на основе некоторого конечного множества элементарных признаков, задаваемых фиксированным набором сенсоров агента. С другой стороны, образ включает и сигналы внутренней среды, так называемый сенсомоторный «рисунок» знака, представляющего предмет внешней среды.
3. Смысл знака задает его связь с внутренней средой агента, с теми сигналами, которые формируются без непосредственного участия внешней среды. К ним в первую очередь относятся потребности агента и его опыт удовлетворения этих потребностей, т.е. опыт выполнения действий и достижения целей.

4. Значение знака является структурой, хранящей согласованную информацию, получаемую и распространяемую за счет коммуникативных действий в группе когнитивных агентов, а также конвенциональные способы ее поиска и извлечения. Значение знака определяет основной набор обобщенных свойств (классов), действий и сценариев, в которые включен данный знак.
5. Знаки связаны друг с другом, образуя систему (семиотическую сеть), которая может быть охарактеризована как согласованное взаимодействие подсистем (каузальных сетей) на базе каждого компонента знака.
6. Система знаков является активной, характеризуясь наличием постоянно меняющегося множества (активных) знаков — фокуса внимания. Правила модификации множества активных знаков называются правилами активации и определяются отдельно для каждого компонента знака. Распространение активности между сетями компонент знаков происходит с помощью функций связывания.
7. Существенным свойством семиотического агента является наличие сети на именах знаков, на основе которой могут быть построены сообщения или высказывания, использующиеся как для поддержания коммуникации агента с другими агентами, так и для формирования объяснения — описания текущей или предсказываемой ситуации. Возможность построения такого сообщения и высказывания по правилам языковой системы, которой пользуется агент, является критерием корректности построенного описания ситуации.
8. Знак представляет как предметы внешней среды, так и явления внутренней среды, при этом обладая одним и тем же строением. Таким образом реализуется мета-уровень представления знаний и возможно моделирование рефлексивной функции поведения агента.
9. Помимо концептуального уровня представления знаний, каждый компонент знака вместе с правилами активации для данного компонента реализуют субконцептуальный уровень представления знаний и обработки информации. Наличие такого уровня для образного и смыслового компонентов знака позволяет моделировать «быстрые» не означаемые агентом операции распознавания и выполнения действий.

Предложенные принципы работы семиотического агента соответствуют теории знаковой картины мира [44], в рамках которой предлагаются модели ряда когнитивных функций человека: планирования поведения [95; 687], распределения ролей [32; 670] и целеполагания [43; 60; 688].

Четырехкомпонентная структура знака определяется фундаментальным разграничением на уровне поступающей и интерпретируемой агентом информации: извне, изнутри и от «значимого другого» [689]. Важным отличием от большинства когнитивных архитектур, в которых предпринимается попытка решения проблемы привязки символов, в том числе является отсутствие явного выделения типов памяти агента. Вся информация о том или ином понятии (объективная, личностная и обобщенная) локализована и доступна за счет обращения к одному имени знака, представляющему данное понятие.

Важной особенностью знакового подхода является возможность использования одних и тех же структур для представления концептуальных рефлексируемых знаний и вспомогательной субсимвольной неозначиваемой информации, которая, однако, может быть выведена на концептуальный уровень (т.е. может быть означена). Это принципиальное свойство реализуется за счет того, что только активация целостного знака как объединения трех компонентов одним именем является процессом вывода информации на концептуальный символный уровень. Участие компонентов знака в процессах распространения активности по соответствующим сетям, которые не приводят к объединению компонентов в знак, также возможно и представляет собой процесс обработки субсимвольной информации, которая не может быть отрефлексирована. При этом объединение компонентов в знак не является случайным, но подчиняется строгим правилам распространения активности [84] и может быть управляемым со стороны семиотического агента, который таким образом динамически регулирует уровень концептуальности представления текущей ситуации. Такой процесс регуляции осуществляется в основном за счет связи знаков на сети имен.

Рефлексия семиотического агента определяется как возможность не только представлять знания о своих собственных знаниях, но и проводить рассуждения, составлять и применять планы мета-когнитивного уровня, в которых действия направлены на изменение внутренней среды агента и его собственных знаний (ментальные действия). К внутренней среде в данном случае относятся и текущие познавательные стратегии, и стратегии составления плана взаимодействия с внешней средой. В такой постановке рефлексия может быть охарактеризована своим уровнем в зависимости от того, какого уровня представления действий включаются во внутреннюю среду агента: представления о предметных действиях во внешней среде, представления о ментальных действиях для изменения предметных планов, представления о ментальных действиях второго уровня для изменения ментальных планов и т.д. [689].

Взаимодействие агента со средой, выработка агентом решений и действий организованы в так называемый когнитивный цикл, который в рассматриваемом случае отличается от стандартных циклов в когнитивных архитектурах отсутствием явного выделения дискретных шагов взаимодействия. Глобальное правило распространения активности при данной формальной интерпретации компонентов знака остается тем же, что и в предыдущих вариантах представления семиотической сети в теории знаковой картины мира: активация одного из компонентов приводит к предактивации (частичной активации) других компонентов знака. Полная активация знака приводит к полному распространению активности по нисходящим связям (в основном, в образном компоненте знака).

6.1.2 Привязка к сенсомоторным данным

Далее будет представлены формальная структура знака и механизмы пополнения его элементов (обучения) за счет методов привязки компонент знака к сенсомоторным

данным. Основным отличием описываемого подхода от классических постановок задачи взаимодействия агента со средой является (в том числе объектной реализации нейросимвольной архитектуры NSLP в главе 4):

- разделение внутренней и внешней сред, с которыми взаимодействует агент,
- учет временной составляющей в формировании представления (образа) агента о внутренней и внешней средах,
- выделение специального канала для информации, поступающей от других носителей знаковой картины мира, находящихся во внешней среде.

Взаимодействие агента A с внешней средой E^{out} и внутренней средой E^{in} происходит за счет получения непрерывного потока первичных сигналов (элементарных признаков) $x_{0:T}^0 = x_t^0, x_{t+1}^0, x_{t+2}^0, \dots$, разделение которых на элементарные моменты времени определяется устройством системы сенсоров агента, и генерации последовательности воздействий (команд) $\Delta y_{0:T}^0 = \Delta y_0^0, \Delta y_1^0, y_2^0 \dots$ агента на внутреннюю среду, дискретизация которых определяется моторными свойствами системы актуаторов (манипуляторов) агента, действующих на внешнюю среду. Верхний индекс 0 здесь указывает на то, что данные элементарные признаки и команды принадлежат к нижнему уровню иерархии обработки поступающих признаков и к нижнему уровню иерархии генерации команд. Далее буду описаны принципы связи признаков и команд в данных иерархиях. Здесь лишь необходимо отметить, что на более высоком уровне предполагается больший масштаб времени. Например, в последовательности $x_t^1, x_{t+1}^1, x_{t+2}^1, \dots$, признак x_t^1 разворачивается в последовательность $x_{T_t:T_{t+1}}^0$.

Таким образом, формируются два потока информации: восходящий, активирующий концептуально более сложные элементы, и нисходящий, активирующий в предсказывающем режиме более простые элементы. Следуя принципу «матрешки» [678], образный компонент каждого знака, ответственная за обработку сигналов связи с внешней средой, должна повторять такую восходяще-нисходящую структуру. В предыдущих работах было предложено использовать для этого *каузальный тензор* [84] — семейство каузальных матриц, каждая из которых, в свою очередь, состоит из последовательности событий, группирующихся локализованные в пространстве и времени признаки нижнего уровня иерархии.

В предыдущих работах [484; 44] предлагался единый формализм для описания всех компонентов знака на основе специального типа семантических сетей — каузальных сетей. В данном разделе диссертационного исследования предлагается развитие сетевого подхода, отражающего специфику работы каждого компонента знака путем использования более специализированных сетевых моделей — отдельных типов каузальных сетей.

Образный компонент знака

Для описания образного компонента будут использоваться *предикторные каузальные сети* — иерархические пространственно-временные схемы на основе марковских цепей, с помощью которых моделируется иерархическая работа процедуры распознавания

наблюдаемого предмета по развернутой во времени последовательности низкоуровневых признаков. Вероятности переходов марковских цепей определяются с помощью описываемых ниже механизмов обучения семиотического агента. Предикторная каузальная сеть является гетерархическим аналогом иерархической временной памяти [279; 280]. Формальное описание сети начинается с определения образа предмета как пространственно-временной схемы $p^i(n) = \langle e_1^i, e_2^i, \dots, e_h^i | v_1^i, v_2^i, \dots, v_k^i \rangle$, где $e_j^i = (x_t^{i-1}(p_1^{i-1}), x_t^{i-1}(p_2^{i-1}), \dots, x_t^{i-1}(p_q^{i-1}))$ — пространственное событие, характеризуемое ссылками на активированные признаки более низкого уровня иерархии p_j^{i-1} и их значениями в текущем событии $x_t^{i-1}(p_j^{i-1})$ (далее действует соглашение о том, что в вектор e_j^i включены только ненулевые элементы), а $v_k^i = (e_{j_k}^i, e_{l_k}^i | z_k^i)$ — переход между двумя событиями $e_{j_k}^i$ и $e_{l_k}^i$, который характеризуется условной вероятностью совершения z_k^i , при этом для каждого события e выполняется условие $\sum_{v_k^i = (e, e_{l_k}^i | z_k^i)} z_k^i = 1$, т.е. сумма вероятностей перехода из этого события равна 1.

Определение 6.1.1 (Предикторная каузальная сеть). *Формально предикторной каузальной сетью $W_p = (T^p, L^p)$ называется ориентированный граф с множеством вершин (узлов сети) $T^p = t_1^p, t_2^p, \dots$ и дуг (ребер) $L^p = (t_1^p, t_{i_2}^p), (t_{i_1}^p, t_{i_2}^p), \dots$. Узел сети t_i^p представляет собой множество пространственно-временных схем $r_j^i(n)$ одного уровня иерархии i и с одним и тем же множеством признаков более низкого уровня иерархии. Направленной дугой $(t_{i_1}^p, t_{i_2}^p)$ связаны два узла сети $t_{i_1}^p t_{i_2}^p$, если в узле $t_{i_2}^p$ есть схема, которая выступает признаком для какой-либо схемы в узле $t_{i_1}^p$.*

Функция распознавания $\Phi^p(x_{0:T}^0, y_{0:T}^0)$ работает за счет распространения активности с использованием правил байесовского вывода по вершинам узлов каузальной сети, в каждом из которых происходит распознавание признаков (образов) из одной предметной области. На первом этапе в момент времени t подсчитывается правдоподобие λ_t каждого события на основе значений входящих в него признаков p_j^{i-1} :

$$\lambda_t(e_k^i) \propto \prod_{j=1}^q x_t^{i-1}(p_j^{i-1}).$$

Вектор значений признаков с нижнего уровня гетерархии подсчитывается с помощью последовательного объединения (сцепления) векторов значений признаков дочерних узлов, которые, в свою очередь, вычисляются с помощью одного из методов динамического программирования:

$$\begin{aligned} x_t^{i-1}(p_j^i) &\sim \prod_{k=1}^h \alpha_t(e_k^i, p_j^i), \\ \alpha_t(e_k^i, p_j^i) &= \lambda_t(e_k^i) \sum_{e_l^i \in p^i(n)} z^i(e_k^i | e_l^i, p_j^i) \alpha_{t-1}(e_l^i, p_j^i), \\ \alpha_0(e_l^i, p_j^i) &= \lambda_0(e_l^i) z^i(e_l^i | p_j^i). \end{aligned}$$

Здесь $z_k^i(e_k^i | e_l^i, p_j^i)$ — вероятность перехода от события e_l^i к событию e_k^i в рамках схемы p_j^i . Приведенные уравнения реализуют восходящее правило распространения активности на каузальной предикторной сети.

Важной особенностью реализации функции распознавания Φ^p является подсчет предсказывающего сигнала, который соответствует распространению активности вниз по иерархии узлов:

$$\begin{aligned}\delta_t(e_k^i) &\sim \prod_{j=1}^K \beta_t(e_k^i, p_j^i), \\ \beta_t(e_k^i, p_j^i) &= \lambda_t(e_k^i) \sum_{e_l^i \in p^i(n)} z^i(e_k^i | e_l^i, p_j^i) \beta_{t-1}(e_l^i, p_j^i), \\ \beta_0(e_l^i, p_j^i) &= \lambda_0(e_l^i) z^i(e_l^i | p_j^i), \\ \hat{x}_t^i(p_j^i) &= \sum_k I(e_k^i) \delta_t(e_k^i).\end{aligned}$$

Здесь K — количество признаков, распознаваемых в рамках данного узла, идентификационная функция $I(e_k^i)$ определяется как

$$I(e_k^i) = \begin{cases} 1, & \text{если } e_k^i \in p_j^i, \\ 0, & \text{иначе.} \end{cases}$$

Представленные уравнения реализуют нынешнее и предсказывающее правила распространения активности на каузальной предикторной сети.

За формирование каузальных схем отвечает модифицированный алгоритм кортикоморфного обучения [242; 84], состоящий из трех основных шагов: формирование пространственного представления (детектирование и обновление событий e_k^i), поиск временных последовательностей (самоформирование образов p_j^i) и выявленной причинно-следственной зависимости (оптимизация пространственно-временных схем p_j^i).

Смысловой компонент знака

Смысловой компонент определяет связь знака с внутренней потребностно-мотивационной сферой семиотического агента и аккумулирует его опыт по удовлетворению той или иной потребности (на нижнем уровне) или достижению той или иной цели на более высоком уровне иерархии действий. В соответствии с предложенной семиотической архитектурой формально смысл знака будет представляться в виде *акторной каузальной сети*, в которой узлами являются множества пар $a^i(n) = (Q_1^i, \beta_1^i, Q_2^i, \beta_2^i, \dots, Q_l^i, \beta_l^i)$, где $Q_j^i(\pi_j^i, x_t^i) \in \mathbb{R}$ — функция полезности, которая по текущим активным признакам x_t^i на уровне абстракции i определяет полезность, выражаемую действительным числом, выполнения действия, задаваемого операционным составом π_j^i при достижении цели β_j^i .

Операционный состав или стратегия действия $\pi_j^i(x_\tau^{i-1}) = \mathbb{P}(a \in a_1^{i-1}, a_2^{i-1}, \dots, a_l^{i-1} | x_\tau^{i-1})$ определяет распределение вероятностей на множестве действий более низкого уровня иерархии в зависимости от текущего наблюдаемого состояния x_t^i и его признаков x_τ^{i-1} . На самом нижнем уровне иерархии находятся моторные команды $\Delta y_{0:T}^0$.

Как это принято в теории обучения с подкреплением, полезность стратегии π_j^i определяется как ожидаемое суммарное вознаграждение, которое получает агент при достижении своей цели $\beta_j^i: Q_j^i(\pi_j^i, x_t^i) \propto \sum_{\pi^{i+1}} \gamma^t r_t$; здесь γ — это дисконтирующий множитель для учета неопределенности реакций среды. В семиотической реализации вознаграждение $r_T = R(y_{0:T}^0, \beta_j^i)$ является некоторой функцией от последовательности внутренних признаков агента $y_{0:T}^0$ и самой цели β_j^i . В данном разделе не будет рассматриваться реализация этой функции и считается, что на каждом уровне иерархии действий в каждый момент времени доступно значение вознаграждения r_t .

Реализация процедуры разворачивания действия $\Phi^a(a(n), x_t^i) = \Delta y_{0:T}^0$ заключается в выполнении следующих шагов на каждом уровне иерархии: идентификация текущей цели β_j^i с учетом цели более верхнего уровня иерархии β_k^{i+1} , выбор оптимальной стратегии с точки зрения полезности $\pi_j^i = \arg \max_{\pi} Q_j^i(x_t^i)$, генерация последовательности операций (операционного состава действия) на основе распределения $\pi_j^i(x_{\tau}^{i-1})$.

Определение 6.1.2 (Акторная каузальная сеть). *Формально акторной каузальной сетью $W_a = (T^a, L^a)$ называется направленный граф с множеством вершин (узлов сети) $T^a = t_1^a, t_2^a, \dots$ и дуг (ребер) $L^a = (t_{i_1}^a, t_{i_2}^a), (t_{i_1}^a, t_{i_2}^a), \dots$. Узел сети t_k^a представляет собой множество пар Q_{ji}, β_{ji} одного уровня иерархии i . Направленной дугой $(t_{i_1}^a, t_{i_2}^a)$ связаны два узла сети $t_{i_1}^a$ и $t_{i_2}^a$, если в узле $t_{i_2}^a$ присутствует стратегия π_j^{i-1} , которая выступает операцией (под-действием) для какой-либо стратегии в узле $t_{i_1}^p$.*

Обобщая сказанное выше, необходимо заметить, что используемая в настоящем разделе структура смысла знака в семиотическом агенте не описывает полностью все аспекты психологического понятия смысла знака, но реализует его ключевые с точки зрения задачи синтеза поведения свойства: включает в себя операционный состав действия и связь с целью агента, для которой представляемый знаком предмет служит препятствием или способом ее достижения. Оценочная часть смысла характеризует опыт агента и определяет силу влияния или степень уверенности в возможности достижения цели за счет данного знака (смысла).

За формирование акторной каузальной сети отвечает один из вариантов иерархического обучения с подкреплением [315; 388] (см. раздел 2.3). В текущей реализации семиотического агента может быть использован метод иерархического актор-критика [143] (см. рисунок 6.3). В алгоритме два базовых шага: оценка действий и мета-действий (действий более высокого уровня иерархии) и улучшение стратегии. Обновления проходят в условиях выбранной цели верхнего уровня β_j^i и соответствующей ей функции $Q_j^i(\pi_j^i, x_T^i) = \sum_t \pi_t^i Q_j^{i-1}(\pi_t^{i-1}, x_t^{i-1})$. На первом шаге считается текущая ошибка предсказанного вознаграждения δ , и на этой основе обновляются полезности Q_j^{i-1} стратегий π_t^{i-1} :

$$\begin{aligned}\delta &= r - Q_j^{i-1}(\pi_t^{i-1}, x_t^{i-1}), \\ \delta &= \delta + \gamma (1 - \beta^{i-1}(x_{t+1}^{i-1})) Q_j^i(\pi_j^i, x_T^i) + \gamma \beta^{i-1}(x_{t+1}^{i-1}) \max \pi' Q_j^i(\pi_j^i, x_T^i), \\ Q_j^{i-1}(\pi_t^{i-1}, x_t^{i-1}) &= Q_j^{i-1}(\pi_t^{i-1}, x_t^{i-1}) + \alpha \delta.\end{aligned}$$

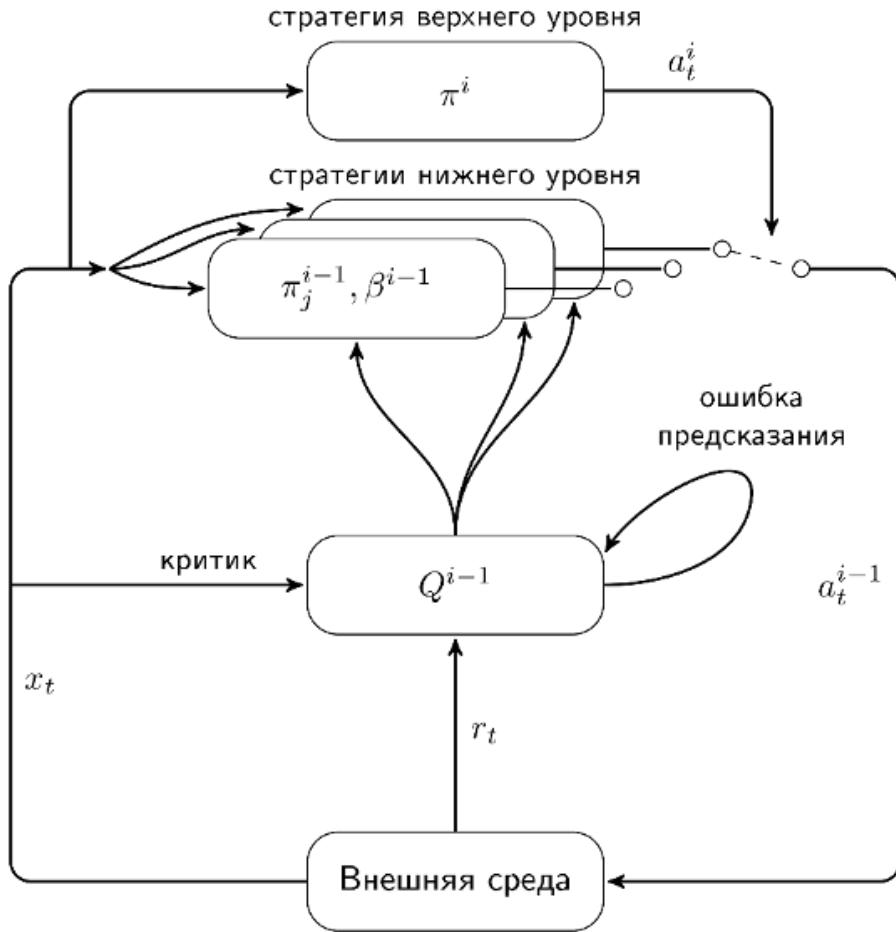


Рисунок 6.3 — Схема обучения акторной каузальной сети с помощью алгоритма иерархического актор-критика.

На втором шаге с помощью градиентного спуска обновляем параметры модели критика ϑ и актора Θ :

$$\vartheta = \vartheta - \alpha_\vartheta \frac{\partial \beta^{i-1}(x_{t+1}^{i-1})}{\partial \vartheta} \left(Q_j^i(\pi_j^i, x_T^i) - \sum_{\pi_j^i} \pi_t^{i-1} Q_j^{i-1}(\pi_t^{i-1}, x_t^{i-1}) \right),$$

где Θ — веса модели актора, ϑ — веса модели критика, α_θ и α_ϑ — размеры градиентных шагов.

Приведенный алгоритм повторяется на каждом уровне иерархии действий и целей семиотического агента.

6.1.3 Сценарий в картине мира

Компонент значения знака

Значение концентрирует в себе обобщенные в коллективе агентов общие знания в виде «словарной» информации. В семиотическом агенте, синтезирующем рациональное поведение, эта информация структурируется по так называемым *сценариям*, которые используются для объяснения происходящих во внешней среде изменений. Множество сценариев образует *сценарную каузальную сеть*. Ядром каждого шага сценария является предикатно-ролевая схема [672] $m^i = \langle n_1^i, n_2^i, \dots, n_k^i | (m_1^i, z_1^i), (m_2^i, z_1^i), \dots, (m_k^i, z_1^i) \rangle$, где m_j^i — ссылка на значение знака-роли, количество которых обычно фиксировано, невелико и определяется структурой используемого в коллективе системой языка формирования сообщений (например, объект, инструментив и т.д.), z_1^i — это действительное целое число от 0 до 1, характеризующее значимость данной роли в структуре значения текущего знака, а n_j^i — соответствующая роли m_j^i ссылка на категориальный класс, заполнитель этой роли. Если, например, $m_1^i = m_2^i = \dots = m_k^i$ и являются ссылкой на значение знака «под-класс», то все n_j^i — экземпляры этого класса, которые могут быть в свою очередь классами для менее абстрактных понятий. Шаги сценария связаны друг с другом причинно-следственными отношениями, которые могут быть представимы в виде стандартного правила «если-то»: если выполнен первый шаг (заполнены все значимые роли значения), то необходимо должен выполниться следующий шаг, а все остальные невозможны. В архитектуре семиотического агента такая необходимость реализуется с помощью активации и деактивации соответствующих знаков.

Определение 6.1.3 (Сценарная каузальная сеть). *Формально сценарной каузальной сетью* $W_m = (T^m, L^m)$ называется направленный граф с множеством вершин (узлов сети) $T^m = t_1^m, t_2^m, \dots$ и дуг (ребер) $L^m = (t_{i_1}^m, t_{i_2}^m), (t_{i_1}^m, t_{i_2}^m), \dots$ двух типов. Узел сети t_k^m представляет собой предикатно-ролевую схему m^i . Направленной дугой $(t_{i_1}^m, t_{i_2}^m)$ первого типа связаны два узла сети $t_{i_1}^m$ и $t_{i_2}^m$, если в узел $t_{i_2}^m$ входит значение $t_{i_1}^m$ в качестве знака-роли, либо заполнителя этой роли. Направленной дугой $(t_{i_1}^m, t_{i_2}^m)$ второго типа (причинно-следственной) связаны два узла сети $t_{i_1}^m$ и $t_{i_2}^m$, если они являются соседними шагами одного сценария.

В теории знаковой картины мира разработаны методы иерархического планирования поведения [670], в которых план представляет собой знак со следующими компонентами: образ — это последовательность ситуаций (этапов плана) со своими признаками и объектами, смысл — это операционный состав стратегии достижения цели плана. Применение операции, составляющей план, в текущей ситуации, задаваемой знаком с образом, включающим все основные компоненты (предметы) текущего окружения агента, приводит к формированию (предсказанию) следующей ситуации.

В данном разделе сценарии будут представлены как обобщение множества планов достижения одной цели, в которых ролями служат формальные этапы плана (обычно задаются роли для трех-четырех этапов), а категориями заполнителями — значения (также обобщенные) предметов, являющихся частью этапов плана. Сценарий выделяется на сценарной каузальной сети тем, что на некотором уровне абстракции, ссылки заполнителей ролей ведут не к абстрактным классам, а к более конкретным объектам и подклассам. При этом для того, чтобы сценарий оставался согласованным, заполнителем семантически эквивалентных ролей для разных этапов плана должны выступать один и те же подклассы.

Формально это записывается следующим образом (далее везде значимость ролей в действии z_j^i будут опущены в предположении, что они равны 1).

Определение 6.1.4 (Простой сценарий). *Простой сценарий Σ' — это структура на сети значений, которая представляет собой три этапа $m = n_1, n_2, \dots, n_k$ «начальный этап», «средний этап», …, «заключительный этап», где n_i — ссылки на значения знаков-действий, в которых в значениях $m^i = \langle n_1^i, n_2^i, \dots, n_k^i | m_1^i, m_2^i, \dots, m_k^i \rangle$ заполнителями одних тех же ролей выступают одни и те же «простые» знаки (с пустыми, элементарными значениями): $n_1^i = n_1^j$, если $m_1^i = m_1^j$.*

Так как значения более низкого уровня иерархии на сценарной каузальной сети интерпретируются как семантические валентности (роли) действий или соответствующих им в языке предикатных слов, то можно сказать, что сценарий — это последовательность предикатных слов, которые связаны одними и теми же ролями [666]. В отличие от обычного значения знака предмета, для которого семантическими ролями обычно являются абстрактные категории, такие как «субъект», «инструментив» и т.д., для этапов сценария ролями являются менее абстрактные категории, такие как, например, «человек» вместо «субъекта» или «столярный инструмент» вместо «инструментива». Обычно такими ролями являются значения предметных знаков, для которых более высокий по уровню иерархии знак являются классом. При этом каждому этапу сценария можно поставить в соответствие такой узел на сценарной каузальной сети, представляющий абстрактное действие, для которого ролевой состав совпадает с ролевым составом этапа сценария с точностью до отношений «класс-подкласс».

Определение 6.1.5 (Сложный сценарий). *Сложным сценарием Σ называется множество простых сценариев $\{\Sigma'_j\}$, имеющих одно и то же начало, т.е.*

$$\Sigma'_i = \langle n_1^i, n_2^i, \dots, n_l^i | m_1^i, m_2^i, \dots, m_k^i \rangle, \Sigma'_j = \langle n_1^j, n_2^j, \dots, n_q^j | m_1^j, m_2^j, \dots, m_k^j \rangle, \\ n_1^i = n_1^j, n_2^i = n_2^j, \dots, n_k^i = n_k^j \text{ и } m_1^i = m_1^j, m_2^i = m_2^j, \dots, m_k^i = m_k^j.$$

Фиксация того или иного уровня абстракции для конкретного плана происходит за счет обновления сети смыслов после получения результатов выполнения плана. В простейшем случае можно считать, что при успешном выполнении плана агент получает вознаграждение +1, а при неудаче -1. Одним из перспективных алгоритмов обновления сети может служить имитационное обучение, но в текущем варианте реализации процедуры адаптации

сценария поведения пользователя применяется иерархическое обучение с подкреплением (см. подробнее далее).

Использование сценария при планировании поведения агента

В соответствии с [60] процесс планирования в знаковой картине мира реализуется с помощью МАР-алгоритма и идет в обратном направлении: от конечной ситуации к начальной. В соответствии с постановкой задачи планирования, представленной в разделе 1.2, на вход алгоритма поступает описание задачи

$$T = \langle N_T, S, S_{start}, S_{goal} \rangle,$$

где N_T — идентификатор задачи, S — множество знаков семиотического агента, S_{start} — начальная ситуация планирования, в которую входят текущие активные знаки фокуса внимания агента, S_{goal} — целевая ситуация, включающая знаки — индикаторы достижения цели.

Результатом МАР-алгоритма является план $Plan = \langle \pi_{plan}, S_1, S_s, \dots, S_T \rangle$ — стратегия π_{plan} выполнения операций агентом — операционная часть личностного смысла и последовательность длины T этапов плана S_i , в которые входят знаки, образы которых являются ключевыми элементами данного шага и которые служат реперными точками при проверке выполнимости плана. При этом ситуация S_i является результатом выполнения i -го действия (операции) стратегии π_{plan} , $S_1 \subseteq S_{start}$ — начальная ситуация, $S_{goal} \subseteq S_T$ — целевая ситуация. Сразу же после построения корректного плана агентом начинается его выполнение. Под выполнением плана понимается генерацию всей иерархии действий вплоть до моторных команд с помощью функции разворачивания Φ^a для каждого этапа плана. После успешного выполнения плана фиксируется достижение цели, обновляются оценки полезности в соответствии с принципом работы каузальной акторной сети и сохраняются прецеденты совершения действий на каждом уровне иерархии акторной сети.

Процесс планирования является иерархическим и состоит из повторения МАР-итерации, включающей в себя четыре этапа:

S-этап — поиск прецедента совершения действий в текущей ситуации S_i ,

M-этап — поиск сценариев (простых или сложных) на сценарной каузальной сети,

A-этап — генерация смыслов, т.е. операционного состава стратегии π_{plan} , соответствующих найденным значениям,

P-этап — построение новой текущей ситуации S_{i+1} по множеству признаков условий найденных действий.

Использование сценария предполагается на М-этапе, когда на j -ой итерации осуществляется поиск предикатно-ролевой схемы m такого действия (простого или сложного сценария), которое бы с помощью функции связывания $\Psi_m^a(m(n)) = a(n)$ было сопоставлено с такой стратегии π_j^i , которая являлась бы выполнимой, т.е. $Q_j^i(\pi_j^i, x_t) > 0$. Реализация

функции Ψ_m^a заключается в подстановке знаков, входящих в текущую ситуацию S_i на места ролей сценария $t(n)$. Разные варианты подстановки приводят к ветвлению процедуры планирования и возможности выбора нескольких возможных планов, которые могут отличаться и по длине, и по составу действий.

6.1.4 Модельный пример

Для иллюстрации работы представленной выше архитектуры семиотического агента были проведены модельные эксперименты по автоматическому формированию сценариев поведения и сохранению опыта выполнения действий в среде с сенсорными данными в виде изображений. Основной целью данного примера является демонстрация работы семиотического агента в постановке задачи обучения с подкреплением с использованием автоматически формируемой агентом модели среды. В качестве среды была взята игра Breakout из набора игр Atari [500], являющаяся стандартной средой, используемой в задачах обучения с подкреплением (см. раздел 1.1).

Цель агента в среде — зарабатывать очки, сбивая небольшим упругим шариком объекты в верхней части экрана. Среда была выбрана конфигурируемой — в ней есть возможность изменять количество сбиваемых объектов-блоков, количество используемых шариков и т.д. Управление движением шарика осуществляется за счет подстановки под него подвижной каретки, за перемещение которой отвечает сам агент. Агенту не предоставляется априорная информация о правилах игры, вариантах реакции среды и условиях начисления очков. В выбранном нами варианте среды при взаимодействии шарика со стеной добавлялся небольшой шум к углу отскока. Правила начисления очков (функция вознаграждения R) составлены следующим образом: +1 за сбитие объекта, -1 за падение шарика мимо платформы, 0 в остальных случаях. Длина эпизода T взаимодействия агента со средой ограничена 500 шагами, максимальное значение вознаграждения за эпизод — 36.

Реализация архитектуры семиотического агента для данного примера выглядит следующим образом. Входной информацией для агента являются цветные изображения — снимки экрана симулятора. Вектор элементарных признаков $x_{0:T}^0$ — это пиксели изображения в области поля зрения агента. Каузальная предикторная сеть W_p в данном эксперименте устроена следующим образом. Семиотический агент способен выделять M различных типов объектов (образов знаков предметов верхнего уровня). Элементарной сущностью является пиксель картинки, а i -ым атрибутом сущности — индикатор соответствия данного пикселя объекту i -ого типа. Всего N сущностей — по размеру фокуса внимания агента. Все сущности пронумерованы в соответствии с координатами их пикселей. В момент времени t по изображению x_t^0 , пришедшему из среды, формируется наблюдаемая ситуация S_t — множество образов наблюдаемых предметов p_i . Количество предметов в фокусе внимания фиксировано и включает центральную сущность и еще R соседних с ней сущностей. R — 1 определяется размером окна зрения агента. В текущей реализации агента использовалась

предобученная функция распознавания Φ^p , которая позволяет получить информацию о принадлежности каждого пикселя определённому образу предмета (см. рисунок 6.4) при этом количество возможных классов объектов также было фиксировано: каретка, шарик, сбиваемые блоки и стена.

Каузальная акторная сеть W_a в проведенном эксперименте в операционном плане состоит из одного уровня иерархии, на котором действия являются элементарными моторными командами $y_{0:T}^0$ по управлению кареткой в среде: не двигаться, сдвинуться влево, сдвинуться вправо. Через a_t обозначается действие, совершенное в момент t . $Q_j(y_j, S_t)$ — оценки полезности действий представляют собой таблицы скалярных величин, которые обновляются в соответствии с правилами обучения критика. Функция развертывания действия Φ^a ввиду наличия только одной цели β (уничтожение всех блоков) и одного уровня иерархии в данном примере не использовалась.

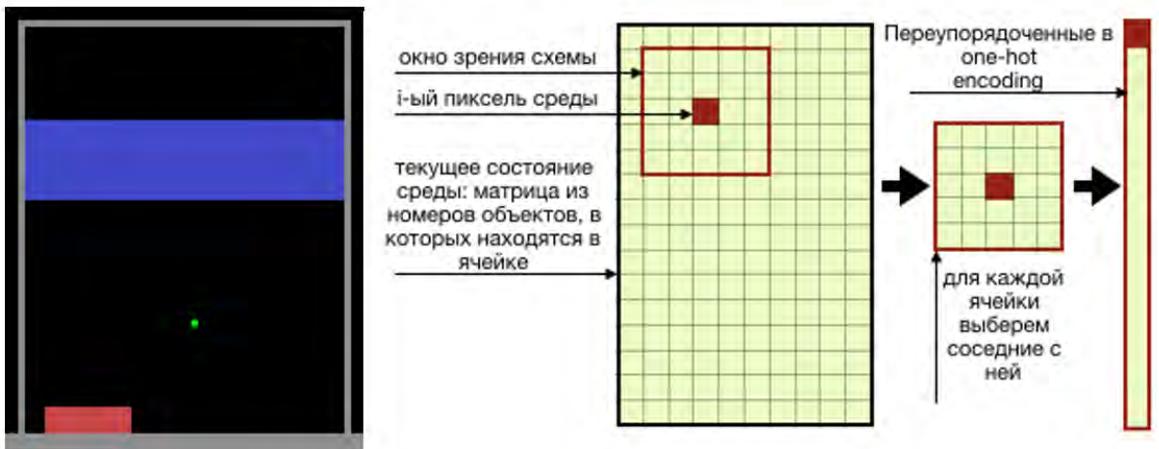


Рисунок 6.4 — Пример расположения рецептивного поля агента и схемы кодирования признаков для функции распознавания Φ^p .

Каузальная сценарная сеть W_m состоит из набора сценариев, которые формируются и обновляются в ходе взаимодействия агента со средой. Пусть M — весь набор сценариев агента, $M_i = m_i^j$ — матрица таких сценариев для предсказания i -ого значения признака некоторого объекта с образом p_j . При формировании сценариев использовалось специальное представление среды S_t , которое складывается из S_t, S_{t-1} и действия a_t [57]. Для предсказания появления или исчезновения предметов в фокусе внимания агентом строились два типа сценариев — формирующие сценарии M^+ и разрушающие M^- . Для прогнозирования следующего состояния S_{t+1} вначале формировались изменения состояния двух типов:

$$\Delta_i^+ = \overline{S_t M_i^+} \mathbf{1}, \Delta_i^- = \overline{S_t M_i^-} \mathbf{1},$$

а затем определялся состав состояния $S_{t+1} = S_t - \Delta_i^+ + \Delta_i^-$. Вознаграждение, получаемое агентом, также кодировалось в сценариях и предсказывалось следующим образом:

$$r_{t+1} = \mathbf{1}^T \overline{\overline{X}_t R} \mathbf{1}.$$

Во время взаимодействия со средой агент сохраняет уникальные цепочки действий в памяти прецедентов. Корректность набора сценариев M проверяется по новым наблюдениям, когда выявляются сценарии, дающие ложноположительные предсказания. В данном примере в результате планирования агент формировал такую последовательность действий, которая приводит его к положительному вознаграждению. Процесс планирования в данном примере немного отличался от стандартной реализации МАР-планировщика и состоял из трех этапов:

1. Прямой проход: строится факторный граф потенциально достижимых узлов сценарной сети W_m .
2. Выбирается набор целевых узлов вознаграждения.
3. Выбирается последовательность действий, которые активируют целевой узел.

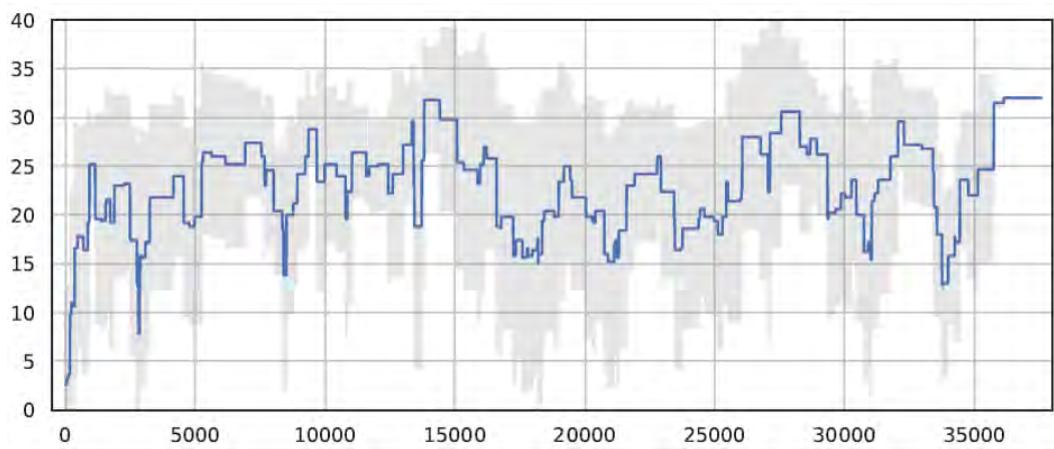


Рисунок 6.5 — Получение вознаграждения семиотическим агентом с использованием сценариев реакции среды (по горизонтальной оси — кол-во шагов, по вертикальной — скользящее среднее по пяти шагам).

В данной среде агент показывает адекватную игру с динамическим формированием сценариев и без изначальных знаний о правилах игры (см. рисунок 6.5). В качестве метрики качества и эффективности работы агента использовалось среднее вознаграждение, которое агент получал по 5 различным эпизодам взаимодействия со средой. Отличительной особенностью сценарной сети является возможность организации эффективного переноса обучаемой модели в среды с аналогичной динамикой. Оценка на окружение с двумя мячами без дополнительной подготовки показала средний балл, так же близкий к 36, что означает, что агент эффективно обобщает правила игры и может взаимодействовать с обоими шариками.

В данном разделе была представлена семиотическая реализация архитектуры NSLP когнитивного агента. Был развит сетевой подход к модели знаковой картины мира. Кроме того, были предложены новые формулировки строения компонентов знака и даны формулы вычисления правил распространения активности по предикторной и акторной каузальным сетям. Было дано оригинальное определение сценария, как основной составляющей сценарной каузальной сети и определена роль сценария при планировании поведения. В экспериментальном исследовании была рассмотрена простая реализация семиотического

агента, который успешно формировал сценарии взаимодействия со средой и использовал их для генерации успешного поведения в базовой игровой среде.

6.2 Когнитивные особенности планирования поведения с языковыми моделями

Построение картины мира — одна из ключевых когнитивных задач, которую приходится решать когнитивным агентам, формирующим свое поведение во внешней среде. Для ее решения необходимо, во-первых, выбрать подходящий способ представления знаний о внешней среде [407; 687], а также о возможностях самого агента в этой среде, что позволит эффективно реализовать методы планирования поведения и предсказания последствий действий. Во-вторых, картина мира агента должна формироваться и обновляться в процессе обучения с использованием сенсорной информации из внешней среды [298]. Одной из ключевых проблем интеграции методов представления знаний и обучения модели мира является проблема привязки символов [270] (в контексте нейросимвольной интеграции), которые используются в задаче планирования поведения для описания действий и их результатов.

Концепция знаковой картины мира (ЗКМ) [42; 44] предполагает формализацию понятия «знака» [302; 505], который опосредует некоторый объект или процесс реального окружения в модели мира агента и обладает компонентной структурой, используемой для решения задачи привязки символов. В этом случае имя знака является частью некоторой языковой системы, которая сама по себе является не только инструментом и регулятором организации коммуникации агентов, но и источником конвенциональных знаний о внешней среде и о самих агентах. Как уже было продемонстрировано разделе 6.1, реализация картины мира когнитивного агента на основе теории ЗКМ способствует созданию так называемых систем воплощенного искусственного интеллекта, которые декларируют важность и необходимость учета специфики взаимодействия агента с окружающей средой для формирования и реализации эффективного плана действий [60].

Основными компонентами знака, помимо имени, являются значение, смысл и образ. Образ — это набор признаков, необходимых для выделения опосредованного объекта из множества остальных в текущей ситуации. Формирование образа может происходить в самоконтролируемом режиме и реализуется современными нейросетевыми кодировщиками [542], в том числе и объектными [471] (см. разделы 4.1 и 4.2). Значение — это набор согласованных в группе агентов правил взаимодействия и использования объектов, опосредованных агентом. Значение знака, по сути, является частью самой языковой системы, в которую входит имя знака. В данном разделе диссертационного исследования предлагается проверить возможность использования современных нейросетевых моделей языка в качестве системы значений, или сигнификатов [376], как вариант технического решения проблемы

бифункциональности слова. В конечном счете, значение — это реализация значимых признаков в конкретной ситуации, и оно может быть смоделировано в процессе обучения и функцией значения.

Моделирование языковой системы для полноценной реализации и функционирования знаковых компонентов до сих пор являлось ключевой проблемой при построении когнитивных агентов с ЗКМ. Однако появившиеся в последнее время большие (более 100 миллиардов обучаемых параметров) предварительно обученные нейросетевые модели языка [152; 373] позволяют рассматривать их как аппроксимацию языковой системы. Использование при обучении таких моделей огромного количества текстовых данных, собранных из всевозможных источников в различных областях, включая повседневную жизнь реальных носителей языка, позволяет говорить о некотором приближении к концепции универсального общего знания (здравого смысла). Такие модели могут быть использованы не только для генерации ответов на естественном языке, но и в более общем случае для генерации любой последовательности лексем (токенов) в ответ на запрос. Причем, в зависимости от решаемой задачи, выбор таких лексем может быть достаточно велик: от набора действий до набора участков (patches) изображения. Таким образом, предварительно обученные большие языковые модели (БЯМ) можно рассматривать как экспертную систему общего назначения, в которой информация и знания, реализованные в языковой системе, могут быть получены по специальному сформированному запросу (подсказке).

БЯМ в данном разделе рассматриваются как приближение к значению, форма операционализации этого психологического понятия для реализации ЗКМ когнитивного агента. Однако вопрос о том, являются ли сущности, над которыми оперируют БЯМ, концептами в психологическом смысле, остается открытым. Опора на концептуальные схемы рассуждений позволяет повысить безопасность моделей при выполнении сложных задач. Для решения этого вопроса предлагается использовать стандартные психодиагностические методики для оценки качества выполнения моделями заданий на понятийное мышление. Эта гипотеза проверяется путем проведения экспериментов на наборе данных, адаптированном для студентов, на основе психологических методик Р. Кэттелла и С.Я. Рубинштейн и сравнения эффективности каждой модели. По результатам этих экспериментов показано, что в ответах на стандартные задания на понятийное мышление можно выделить несколько типов модельных ошибок и оценить их тип в соответствии с классификациями искажений понятийного мышления, принятыми в культурно-историческом подходе в психологии. Это позволяет использовать психодиагностические методики не только для оценки эффективности моделей, но и для разработки обучающих процедур на основе таких заданий.

6.2.1 Психологические подходы к мышлению и речи

Многие авторы [375; 578] сталкиваются с необходимостью рассмотрения психологических подходов для объяснения формирования навыков и использования языка для построения

внутренней (на этапе формирования запроса) и внешней коммуникации агента для достижения цели при выполнении действий. Так, создатели системы SayCan [233] отмечают, что возникают трудности в адаптации работы БЯМ для задачи моделирования рассуждений и составления описаний действий на основе инструкций для робототехнической системы. Эти трудности приводят к высокоуровневым рассуждениям о сложных задачах и разнообразию плана выполнения действий, сложных для реализации агентом с ограниченным набором навыков, с учетом особенностей функционирования конкретной среды. Опора на понятийные схемы рассуждений позволяет повысить успешность моделей при решении такого рода сложных задач.

В качестве методологических средств, объясняющих связь между формированием понятий и их использованием в контексте изучения речи, можно рассматривать психологические концепции из культурно-исторической психологии Л. Выготского [662] и исследования Ж. Пиаже [495]. Это позволяет многоаспектно рассматривать изучаемые модели воплощенных когнитивных агентов. Вопрос использования моделей мышления для разработки и валидации БЯМ получил развитие в рамках методологии теории разума (Theory of Mind, ТоМ) [221]. В последние годы большое внимание уделяется использованию психологических моделей мышления и рассуждений, причем акцент делается не на использовании формальных понятий и рассуждений, а на реализации коммуникативных функций и решении повседневных задач. В работе Т. Ульмана [627] ставилась задача рассмотреть процессы мышления на основе решения повседневных атрибутивных задач. В экспериментах требовалось предсказать, где находится предмет в ситуации различных словесных описаний ящиков с разным содержимым (попкорн и шоколад). В основу модели рассуждений была положена ТоМ, предложенная Д. Деннеттом. В основе этой теории лежит предположение о скрытости психических процессов и необходимости определения причин поведения в ситуации ложных ожиданий. Как отмечает Деннетт, такой подход важен для определения причин поведения, но он не всегда справедлив для решения прямых рациональных задач, к которым относятся и формальные задачи рассуждения на основе понятий и обобщений.

Альтернативные теории представлены более широким рассмотрением круга задач, которые решаются в процессе взаимного развития речи и мышления (как это было предложено в работах Пиаже и Выготского по исследованию развития речи и мышления в онтогенезе). Выготский отмечал важность связи мышления и речи, а также необходимость рассмотрения значения слова в смысловом единстве. С точки зрения психолингвистики [505], слово (имя) выполняет две функции: индикативную, указывающую на соотнесенность с объектами в предметном поле, и сигнификативную, связанную с пониманием смысла и функционального значения. В процессе развития человека значение слова расширяется, дополняется опытом и определением дополнительных, существенных признаков благодаря общению в социальной среде. Выготский считал, что обобщение значения слова, по сути, и есть процесс формирования понятия. Для психологов было важно определить функциональное значение вербальных знаков и сам процесс формирования понятий. Характер стратегии решения этой задачи позволил Выготскому

сформулировать последовательность этапов (шагов) генерализации значений слова в понятие и выделить этапы перехода от предметной связи слова с предметным миром к раскрытию сигнификативной функции.

Выготский выделяет три этапа формирования рациональных понятий:

Синкреты — слово (имя), обозначающее объекты, сходные по некоторым субъективным характеристикам и логически принадлежащие к разным классам.

Комплексы не раскрывают всей полноты отношений признаков в группе, обозначаемой одним словом. В данном случае слово — это не понятие, а название нескольких объектов, слово-«кличка».

Истинные понятия — на этом этапе решаются две задачи: переход от отражения действительности в ее ситуативных зрительных образах к отражению в понятиях, правилах и переход от простого воспроизведения репрезентаций к умственным действиям, т.е. к решению задач, формулированию и проверке гипотез.

Развитие понятийного мышления, тесную взаимосвязь этапов мышления и речи отмечал французский психолог Пиаже, имевший как общие, так и дискуссионные позиции с Выготским по социокультурным принципам развития психики человека. Пиаже выделял этапы развития интеллекта и поддерживал линию развития мышления от действий (операций) к формированию словесно-логических связей [495].

Для построения систем, не только использующих компоненты базы слов как набор символов, но и учитывающих многоуровневый характер знаний, можно считать ЗКМ, предложенную в работах Г. Осипова, как основу для разработок систем воплощенного искусственного интеллекта. В ней выделяются три типа картины мира, основанные на различных компонентах знака и опыта: рациональный, основанный на формальных знаниях и обобщениях; обыденный, основанный на сценарных знаниях; и мифологический, основанный на эмоциях и личностных смыслах.

6.2.2 Постановка психологического эксперимента

В настоящем разделе исследуется то, как современные БЯМ справляются с решением задач в области представления картины мира и с построением понятий. Для этого предлагается использовать стандартные психодиагностические методики для оценки качества выполнения БЯМ заданий на понятийное мышление и сравнения их эффективности. В основу экспериментального материала положены базовые методики изучения истинных понятий (формальных понятий) и выделения существенных признаков. Аналогичные экспериментальные методики с использованием психодиагностических инструментов проводились в современных работах для оценки эффективности моделей ChatGPT. М. Косински рассматривает использование классических задач для проверки эффективности нескольких языковых моделей [362]. Он указывает на улучшение современных моделей при решении задач на выявление ложных убеждений, что во многом

связано с характером самих задач, которые основаны на повседневном опыте и не требуют перехода на уровень абстрактных понятий. Косвенно с этим столкнулись при последующем генерировании авторы работы [620], которые отметили проблему того, что БЯМ могут быть потенциально вредны, демонстрируя манипулятивное и нарциссическое поведение. Использование готовых психологических инструментов для оценки эффективности каждой из моделей позволяет построить новую версию экспертной проверки результатов ответов БЯМ на основе соотнесения с готовыми ответами и типами ошибок в заданиях. В данном исследовании предлагается сосредоточиться на понятийных заданиях, сочетающих процесс работы с языковыми моделями и рассуждения.

Была разработана методика проведения исследования, состоящая из нескольких этапов:

1. Выбор психологического показателя для оценки успешности деятельности (например, соотношение употребления слова и уровня обобщения и градации этого признака).
2. Подбор психологических методик, которые используются в диагностике этих признаков (например, изучение высказываний или классификация при решении задач на вербально-логическое мышление).
3. Составление массива вопросов на основе заданий психологических методик (например, перевод заданий в диалоговую форму для БЯМ).
4. Проведение экспериментов с несколькими итерациями предъявления задач для каждого из БЯМ (например, предъявление вопроса из набора данных десять раз для оценки эффективности работы модели над задачей и процента успеха).
5. Качественная или количественная оценка успешности решения задач, связанных с психологическим феноменом.

В проведенных экспериментах модели оценивались в режиме без дополнительных примеров (zero-shot), но для дальнейших экспериментов можно использовать режим с несколькими примерами или режим дообучения.

Подобная схема использования психологических методик в качестве основы для набора данных была предложена в работе Косински, который взял 40 классических задач на ложное убеждение, широко используемых для тестирования ТоМ у человека [362], и показал эффективность решения задач распознавания убеждений в повседневных ситуациях для различных версий моделей семейства GPT (GPT-3 [376], GPT-3.5, GPT-4 [291]).

Набор данных для проведения тестирования

Для проведения эксперимента, направленного на изучение использования БЯМ для решения задач, основанных на использовании истинных понятий и процессов рассуждения с учетом цели задачи, были использованы две валидные психологические методики, выделяющие параметры формирования понятия, обобщения и сохранения цели рассуждения.

Первая методика — это задания из теста интеллекта, взятые по шкале «Scale B 16 PF» Кэттелла [170]. Методика данного опросника основана на лексической гипотезе. Задания из теста преобразованы в диалоговую форму вопросов и представлены на английском языке, поскольку для ведения диалога рассматриваемые БЯМ обучены на корпусе английского языка. Такие лингвистические задания с концептами, с одной стороны, ориентированы на проверку правильности ответа и принципа объяснения вывода. С другой стороны, это проверка использования смыслового сходства концептов для объяснений на основе БЯМ, а также принятие рассуждения на основе направленности когнитивной задачи, включающей выделение целевого компонента ответа на вопрос не как набора слов, а как процесса рассуждения с определением данных условий и задач для решения.

Initial question	Modification
<p>Which of these words does not fit the other two:</p> <ol style="list-style-type: none"> 1. candle 2. moon 3. lamp 	<p>Which of the words candle, moon, lamp do not fit the other two?</p>
<p>The word "Shovel" refers to the word "dig" as the word "knife" refers to the word:</p> <ol style="list-style-type: none"> 1. sharp 2. cutting 3. sharpening 	<p>What do people do with a knife to cook food: cut, sharpen, kitchen?</p>

Рисунок 6.6 — Примеры из «Scale B 16 PF» опросника R.Cattell и их модификация для запроса БЯМ.

Второй класс методик, которые легли в основу составления вопросов, был взят на основе объединения слов по классам на основе выделения существенных признаков понятий. За основу взят тест Рубинштейна «Выделение существенных признаков» [680]. Задания также были переведены на английский язык и представлены в виде вопросов. Методика этого теста выявляет логику суждений, а также способность сохранять направленность и устойчивость метода рассуждений при решении длинной серии однотипных задач. Подобный класс методик рассматривается в клинической психологии.

В соответствии с методологией исследования была составлена база вопросов на основе двух методик: опросника Кэттелла «Scale B 16 PF» (взято девять вопросов, связанных с понятиями и их значениями) и методики Рубинштейн «Выделение существенных признаков» [680] (выбрано пять вопросов). Небольшое количество заданий объясняется сложностью автоматического сопоставления ответов модели с ответами анкеты. Автоматизация процесса верификации и увеличение числа задач — важное направление возможных дальнейших исследований. Примеры вопросов и их модификаций для «Scale B 16 PF» опросника РКэттелла и методики «Выделение существенных признаков» Рубинштейн приведены на рисунках 6.6 и 6.7 соответственно.

Initial question	Modification
Choose two words from the list that are most related in meaning to the word "Garden": plants, gardener, dog, fence, land.	From the words plants, gardener, dog, fence, land, which two words are closest in meaning to the word "Garden"?
Choose two words from the list that are most related in meaning to the word "Reading": eyes, book, glasses, text, word.	From the words eyes, book, glasses, text, word, which two words are closest in meaning to the word "Reading"?

Рисунок 6.7 — Примеры из методики Рубинштейн «Выделение существенных признаков» и их модификация для запросов к БЯМ.

Рассматриваемые языковые модели

Большинство современных исследований в области БЯМ [128; 166; 362; 478; 627] посвящено моделям с большим числом параметров, таким как GPT-3 [376], FLAN [263], PaLM [487], ChatGPT¹ и др. Отчасти это связано с тем, что такие модели являются наиболее эффективными реализациями БЯМ, а отчасти с тем, что некоторые свойства, например, цепь рассуждений (chain of thought) [173], начинают оказывать существенное влияние только при использовании большого числа параметров. Однако такие модели либо вовсе отсутствуют в открытом доступе, либо имеют ограниченный доступ. Исследование БЯМ с меньшим числом параметров видится важной научной и практической темой, поскольку они являются преимущественно общедоступными.

Для исследования построения истинных понятий БЯМ были выбраны следующие модели семейства GPT с возрастающим числом параметров: GPT-J 6B² (GPT-J), GPT-NeoX-20B [292] (NeoX) и Bloom 176B [156] (Bloom).

GPT-J 6B — это GPT-2-подобная [377] языковая модель с 6 миллиардами параметров, обученная на наборе данных Pile [612]. GPT-NeoX-20B — GPT-3- подобная [376] языковая модель с 20 миллиардами параметров, также обученная на наборе данных Pile. Bloom представляет собой GPT-3-подобную языковую модель с 176 миллиардами параметров, обученную на 46 различных языках и 13 языках программирования. Для всех моделей были использованы реализации из репозитория Hugging Face³. Для формирования релевантного количества ответов было сделано по десять презентаций каждого из 14 вопросов для каждой модели.

¹<https://openai.com/blog/chatgpt>

²GPT-J-6B: a 6 billion parameter autoregressive language model <https://github.com/kingoflolz/mesh-transformer-jax>.

³<https://huggingface.co>

Также были проведены пробные эксперименты с моделью ChatGPT, для которой каждый вопрос подавался трижды. Эта модель дает стабильные ответы, что позволило сократить количество запросов до трех вместо десяти.

Для изучения формирования понятийных ответов у БЯМ была использована следующая экспериментальная схема. Все модели тестировались в режиме без дополнительных примеров (zero-shot), т.е. в качестве входных данных модели использовалось только модифицированное задание в виде вопроса на английском языке без дополнительных подсказок. Для проверки устойчивости модели каждое из 14 заданий предъявлялось модели десять раз, и ответы модели фиксировались. Для доступа к модели использовался стандартный интерфейс запросов для репозитория Huginface. Оценка ответов модели производилась экспертом-психологом вручную в соответствии с ответами, представленными в разработанной методике проведения испытаний. Такой подход был выбран в связи со сложностью автоматизации процесса оценки на предварительном этапе исследования.

6.2.3 Результаты экспериментального исследования

В ходе экспериментов ответы моделей не всегда приводили к правильным решениям и имели ряд искажений. Все ответы можно разделить на несколько категорий в соответствии с типами ошибок из психоdiagностических методик оценки понятийного мышления:

- А — опора на незначительные признаки, латентные признаки.
- В — отсутствие цели решения задачи в нахождении правильного ответа.
- С — правильное описание значений слов из вопроса и отсутствие обобщений на основе существенных признаков понятий.
- D — включение эмоционального компонента.
- I — перенос задачи на другую структуру деятельности, например, решение задач коммуникативного взаимодействия.
- F — бессвязные тексты.

Эти категории ошибок по своему содержанию близки к типам нарушений мышления, диагностируемым в клинической психологии (нарушение мышления в виде редукции или искажения обобщений, динамического и мотивационного компонентов). При анализе БЯМ эти категории позволяют выявить тип ошибки и наметить пути ее решения путем модификации запроса.

Ниже представлены примеры ответов с отмеченными типами ошибок, которые получаются на исследуемых моделях.

A - NeoX: «Рыба» и «грязь» не имеют ничего общего ни с одним из других слов. Все прилагательные относятся к совершенно разным типам метафор. Между ними нет никакой определяющей связи. Этот редукционистский взгляд не сохраняет искусство:

B - Bloom: Какое значение и смысл он будет иметь в предложении? Есть ли среди них те, которые подходят и к слову «несколько», и к слову «большинство»? В чем смысл и значение такого предложения? Ответ: Лучшей альтернативой будет наличие ни одного или не более одного из вариантов. Таким образом, альтернативой может быть следующее предложение: «Не более одного из трех студентов заполнили анкету». Проблема здесь скорее «в мире, для которого решает английский язык», например, отсутствие молекул или атомов, доступных для визуализации, или возможность использовать только грубую оценку, основанную на ограниченной предварительной выборке, или даже то, что сама задача может быть недостаточно конкретизирована, если гипотетически может быть пропущено какое-то новое наблюдение, которое добавит до 10-20% новых точек данных, если это так.

C - GPT-J: Я пытался придумать наиболее правильное слово для каждого из трех слов, не глядя на полный абзац. *A: Свеча-луна-лампа*, потому что они, по-видимому, расположены последовательно или, по крайней мере, в одном и том же месте в начале.

D - Bloom: Это вопрос с подвохом, созданный для того, чтобы заманить в ловушку слабонервных людей с помощью диагностики по тесту характера, и мы пытаемся ответить на него с тех пор, как человек впервые подумал, что он образец в банке.

I - NeoX: Что люди делают с ножом для приготовления пищи: режут, точат, готовят? Что бы это ни было, человек по имени Майк Чепмен за последний год пересчитал ножей больше, чем я могу пересчитать по пальцам обеих рук, и интервью с ним привело меня к выводу. Итак, я начну с краткой молитвы за молодую леди, пока вы читаете это. Она даст вам лучшее представление о том, о чем я пишу. Позвольте, однако, представиться. Индианаполис, штат Индиана, США, нашел меня однажды субботним вечером и спросил, почему я изучаю специфический черный рынок ножей, казалось бы, вне сферы моих интересов и кода района, ха-ха.

Для модели GPT-J наиболее частотные ответы были построены по типу формирования псевдопонятия, где основной ошибкой было нарушение целевого компонента рассуждения. Ответы этой модели по использованию лингвистического поля понятий соответствуют теме, что свидетельствует о корректности решения задач с помощью этой БЯМ. Однако ответ оказывается неверным, так как не сохраняется цель задания на рассуждение — не привлекать дополнительные знания, а сузить их до заданных в вопросе. При этом выбранные критерии небесных тел (солнца и луны) не подобраны по существенным признакам, но лишь иллюстрируют пример псевдопонимания. Критерий объясняется в контексте поиска обобщающего слова, что также поддерживается в контексте использования только слов, но без постановки целевой задачи его выбора. Это подменяет цель ответа и приводит к неправильным решениям задачи. Можно сделать вывод, что эта модель БЯМ опирается на вторичные признаки, а не на значение понятия.

Для Bloom наиболее распространены ответы коммуникативного характера и описания, основанные на латентных признаках из далеких концептуальных классов. Большинство ответов NeoX содержат бессмысленный или эмоционально насыщенный текст. Таким

образом, наиболее близким к решению оказался GPT-J, дающий ответы на каждый из тестов. Эта модель имеет уклон в сторону использования псевдопонятий, основанных на латентных признаках.

Чтобы проверить эффективность оценки работы с понятиями и задачами на рациональное мышление, был поставлен поисковый эксперимент с наиболее известным представителем БЯМ — ChatGPT. Было получено достаточное количество правильных ответов, описывающих критерии рациональности. Например, один из ответов выглядел следующим образом:

Слово «думать» не относится к двум другим словам — «видеть» и «слышать». Под словом «думать» понимается процесс формирования мыслей, рассуждений и суждений с помощью разума. Это внутренняя познавательная деятельность, которая не обязательно связана с внешними органами чувств — зрением или слухом.

Это позволяет оценить эффективность мультидисциплинарного подхода с использованием знакового опосредования как критерия работы с БЯМ. В дальнейшем возможно расширение анализа за счет оценки процента правильных ответов или типов ответов для каждой модели, а также возможен переход к использованию нескольких примеров ответов или дообучения для включения более подробного представления цели в модель рассуждений.

Модели БЯМ в проведенном исследовании давали ответы, содержащие те же классы искажений ответов, что и в ответах на вопросы методики «Scale B 16 PF» Кэттелла. Одним из подходов к проблеме разработки архитектуры воплощенных агентов для решения сложных задач является теория ЗКМ, построенная на основе культурно-исторического и деятельностного подходов. В соответствии с этой теорией при построении картины мира интеллектуального агента предполагается учитывать концептуальный уровень путем рассмотрения категорий значения и смысла. ЗКМ предполагает изучение компонентов знаковой структуры с учетом опосредования объективной реальности и лингвистической составляющей значения слова. Это дает методологические возможности для разработки моделей, учитывающих обе функции слова (имени) и тем самым позволяющих корректно интерпретировать инструкции на естественном языке. Методологически значимой является разработка внутреннего плана целенаправленных действий агентов, поскольку при этом происходит формирование многопланового мышления, непосредственно учитывающего эту информацию и ее второй план — личностные смыслы.

В данном разделе были использованы стандартные психодиагностические методики для оценки сформированности реальных понятий у БЯМ. Результаты показывают, что рассматриваемые модели не справляются с заданиями из психологических опросников и, как уже отмечалось выше, дают ответы, содержащие те же классы искажений ответов, что и в ответах на вопросы методики «Scale B 16 PF» Кэттелла. Таким образом, методологический инструментарий психологических концепций, основанный на теории культурно-исторической психологии, демонстрирует модельные ограничения в использовании имеющихся БЯМ.

В качестве дальнейших исследований, развивающих представленный в данном разделе подход, можно выделить расширение набора данных и типов анкет, автоматизацию

оценки ответов модели, что позволит провести тщательный количественный анализ. В представленном эксперименте для тестирования модели использовался только режим без дополнительных примеров. Однако тестирование в режиме предоставления нескольких примеров решения задачи или с дополнительным дообучением моделей на размеченных данных также является важным направлением исследований.

6.3 Выводы

В данной главе диссертационного исследования была рассмотрена семиотическая реализация архитектуры NSLP генерации поведения когнитивного агента. Проблема привязки символов в контексте нейросимвольной интеграции в семиотическом подходе решается за счет введения специально четырехкомпонентной структуры — знака. Каждый компонент знака (имя, образ, значение и личностный смысл) формализуется специальной графовой структурой — каузальной сетью. Для каждого компонента представлены процедуры по пополнению в процессе обучения и актуализации при построении знакового описания текущей наблюдаемой агентом ситуации за счет специальных процедур активации и связывания.

В контексте данной работы наибольшее значение для реализации механизма одновременного обучения и планирования играют компоненты личностного смысла и значения. Обновление и модификация акторной каузальной сети, формализующей понятие смысла, происходит за счет иерархического обучения с подкреплением. Планирование опирается в первую очередь на компонент значения и формализующую его сценарную каузальную сеть. Описаны понятия простого и сложного сценария, которые используются в алгоритме планирования МАР.

Для демонстрации работы семиотической реализации архитектуры NSLP была выбрана одна из сред Atari, для которой были описаны конкретные реализации акторной и сценарной сетей и процедур их модификации. Проведенные эксперименты показали, что семиотический агент успешно обучается эффективной стратегии поведения в данной среде.

Также в главе были рассмотрены когнитивные аспекты использования предобученных больших языковых моделей в семиотической парадигме. Была предложена методика психологического эксперимента с использованием шкалы Р. Кэттелла и методики С.Я. Рубинштейн. Было рассмотрено несколько языковых моделей (Neo-X, Bloom, Chat-GPT и др.). Наиболее близкой к решению задач рассмотренных методик оказалась модель GPT-J, дающая ответы на каждый из тестов. Эта модель имеет уклон в сторону использования псевдопонятий, основанных на латентных признаках. Данный эксперимент демонстрирует ограничения по использованию современных языковых моделей в качестве высокоуровневых планировщиков в архитектуре NSLP и обосновывает необходимость их дообучения на предварительно собранных данных или в симуляционной среде.

Заключение

В диссертации предложены и теоретически обоснованы новые математические модели, методы и алгоритмы генерации поведения когнитивного агента в динамической среде. В данной работе впервые предложен единый подход к проблеме привязки символов за счет разработки методов нейросимвольной интеграции в обучении и планировании.

Основные результаты работы заключаются в следующем:

1. Была разработана нейросимвольная архитектура управления поведением когнитивного агента, включающая в себя компоненты одновременного планирования и обучения, а также компонент концептуального планирования с использованием языковых моделей.
2. Были разработаны модели и методы интеграции планирования и обучения с подкреплением, в том числе с использованием модели среды, для решения сложных визуальных и векторных задач управления поведением когнитивным агентом, в том числе в многоагентной постановке.
3. Были созданы модели и методы объектно-центричного подхода к представлению сенсорной информации о статических сценах для использования в нейросимвольной архитектуре управления поведением когнитивного агента.
4. Были разработаны модели и методы объектно-центричного обучения с подкреплением с использованием динамической модели среды для интеграции планирования и обучения в нейросимвольной архитектуре управления поведением когнитивного агента.
5. Был усовершенствован ряд существующих моделей и методов обучения с подкреплением на основе модели мира, с моделями внутренней мотивации и с использованием эвристических планировщиков.
6. Была разработана программная реализация системы управления робототехническими платформами с использованием языковых моделей для подзадачи планирования.

Разработан программно-алгоритмический инструментарий, основанный, в том числе на полученных теоретических результатах, для решения задачи генерации действий робототехнической платформой в сложной динамической среде, позволяющий использовать как обучаемые компоненты, так и классические планировочные. Создана экспериментальная программа реализация элементов данного инструментария, использующаяся для решения практических задач управления поведением. Продемонстрировано использование разработанных моделей и методов одновременного планирования и обучения в ряде практически важных робототехнических задачах: навигация мобильной платформы внутри помещений, адаптивное планирование маневров беспилотным транспортным средством, перемещение и манипуляция объектами мобильной платформы по языковым инструкциям.

Разработанные в рамках диссертации методы и алгоритмы использовались для решения следующих прикладных задач:

- автоматическое планирование маршрута и траектории движения функционирующего в составе комплекта аппаратуры управления транспортного средства;
- одновременные локализация и картирование местности для мобильных роботов, функционирующих в разделяемой с людьми среде;
- построение динамической карты проходимости и планирование на ее основе движения мобильных наземных роботов;
- автоматическое управление движением автомобильного транспортного средства в условиях дорог общего пользования;
- управление роботом с модулем планирования по языковым инструкциям для сортировки объектов в помещении с использованием мобильного робота и манипулятора.

Перспективы дальнейшего развития текущих исследований. В качестве одного из основных направлений необходимо указать разработку более эффективных методов объектно-центричного представления информации и объектно-центрических методов обучения с подкреплением на основе модели среды, функционирующие в более сложных и реалистичных средах. Также требуется усовершенствование представленного программно-аппаратного комплекса для более глубокой интеграции с описанными в диссертации гибридными методами обучения с подкреплением и планирования.

В целях дальнейшей проработки нейросимвольного уровня архитектуры NSLP необходимо создавать новые мультимодальные модели привязки действий к подцелям на основе имеющихся моделей привязки текстовых описаний к изображениям (CLIP [400], R3M [517]). Следуя полученным в данном диссертационном исследовании результатам, такие модели также должны быть объектно-центрическими.

Наконец, для улучшения работы концептуального уровня архитектуры NSLP необходимо дообучать большие языковые модели используемые для генерации гипотез, подаваемых планировщику. Известные работы в области дообучения на заранее собранных данных с интерпретируемой как вознаграждение разметкой качества (RLHF [621], DPO [226]) генерируемых ответов модели должны быть расширены на режим дообучения в интерактивной среде. Многообещающее направление работы задают исследования в области обучения мультимодальных моделей мира (Dynalang [397]).

Публикации автора по теме диссертации

1. Applying Vector Symbolic Architecture and Semiotic Approach to Visual Dialog / A. Panov [et al.] // Hybrid Artificial Intelligent Systems. HAIS 2021. Lecture Notes in Computer Science. Vol. 12886 / ed. by H. S. González [et al.]. — 2021. — P. 243—255.
2. Decentralized Monte Carlo Tree Search for Partially Observable Multi-agent Pathfinding / A. Panov [et al.] // Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 38 (AAAI). — 2024. — P. 17531—17540.
3. Fine-tuning Multimodal Transformer Models for Generating Actions in Virtual and Real Environments / A. Panov [et al.] // IEEE Access. — 2023. — Vol. 11. — P. 130548—130559.
4. Forgetful experience replay in hierarchical reinforcement learning from expert demonstrations / A. Panov [et al.] // Knowledge-Based Systems. — 2021. — Vol. 218. — P. 106844. — Publisher: Elsevier B.V.
5. Hierarchical Deep Q-Network from imperfect demonstrations in Minecraft / A. Panov [et al.] // Cognitive Systems Research. — 2021. — Vol. 65. — P. 74—78.
6. Hierarchical intrinsically motivated agent planning behavior with dreaming in grid environments / A. Panov [et al.] // Brain Informatics. — 2022. — Vol. 9, no. 1. — P. 8.
7. Hybrid Policy Learning for Multi-Agent Pathfinding / A. I. Panov [et al.] // IEEE Access. — 2021. — Vol. 9. — P. 126034—126047.
8. Interactive Grounded Language Understanding in a Collaborative Environment: Retrospective on IGLU 2022 Competition / A. Panov [et al.] // Proceedings of the NeurIPS 2022 Competitions Track, PMLR. — 2023. — Vol. 220. — P. 204—216.
9. Interactive Semantic Map Representation for Skill-Based Visual Object Navigation / A. Panov [et al.] // IEEE Access. — 2024. — Vol. 12. — P. 44628—44639.
10. Learn to Follow: Decentralized Lifelong Multi-Agent Pathfinding via Planning and Learning / A. Panov [et al.] // Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). — 2024.
11. Learning embodied agents with policy gradients to navigate in realistic environments / A. Panov [et al.] // Advances in Neural Computation, Machine Learning, and Cognitive Research IV. NEUROINFORMATICS 2020. Studies in Computational Intelligence. Vol. 925 / ed. by B. Kryzhanovsky [et al.]. — Springer International Publishing, 2021. — P. 212—221.
12. Monte-Carlo Tree Search for Multi-Agent Pathfinding: Preliminary Results / A. Panov [et al.] // Hybrid Artificial Intelligent Systems. HAIS 2023. Lecture Notes in Computer Science. Vol. 14001 (HAIS 2023). — Springer Cham, 2023. — P. 649—660. — (Lecture Notes in Computer Science).
13. Multilayer cognitive architecture for UAV control / A. I. Panov [et al.] // Cognitive Systems Research. — 2016. — Vol. 39. — P. 58—72.
14. Neural Potential Field for Obstacle-Aware Local Motion Planning / A. Panov [et al.] // 2024 IEEE International Conference on Robotics and Automation (ICRA) (ICRA 2024). — 2024. — P. 9313—9320.

15. Object Detection with Deep Neural Networks for Reinforcement Learning in the Task of Autonomous Vehicles Path Planning at the Intersection / A. Panov [et al.] // Optical Memory and Neural Networks. — 2019. — Vol. 28, no. 4. — P. 283—295.
16. Object-Centric Learning with Slot Mixture Module / A. Panov [et al.] // The Twelfth International Conference on Learning Representations (ICLR 2024). — 2024.
17. *Panov, A. I.* Symbolic Disentangled Representations in Hyperdimensional Latent Space / A. I. Panov, A. Korchemniy, A. Kovalev // ICLR NeSy-GeMs Workshop (ICLR 2023). — 2023.
18. *Panov, A. I.* A World Model for Actor—Critic in Reinforcement Learning / A. I. Panov, L. Ugadiarov // Pattern Recognition and Image Analysis. — 2023. — Vol. 33, no. 3. — P. 467—477.
19. *Panov, A.* Transfer Learning with Demonstration Forgetting for Robotic Manipulator / A. Panov, E. Aitygulov // Procedia Computer Science. — 2021. — Vol. 186. — P. 374—380.
20. *Panov, A.* Task and Spatial Planning by the Cognitive Agent with Human-like Knowledge Representation / A. Panov, E. Aitygulov, G. Kiselev // Interactive Collaborative Robotics. ICR 2018. Lecture Notes in Computer Science. Vol. 11097 / ed. by A. Ronzhin, G. Rigoll, R. Meshcheryakov. — Springer, 2018. — P. 1—12.
21. *Panov, A.* Approximation Methods for Monte Carlo Tree Search / A. Panov, K. Aksenov // Proceedings of the Fourth International Scientific Conference “Intelligent Information Technologies for Industry” (IITI’19). IITI’19 2019. Advances in Intelligent Systems and Computing. Vol. 1156 / ed. by S. Kovalev [et al.]. — Springer International Publishing, 2020. — P. 68—74.
22. *Panov, A.* Policy Optimization to Learn Adaptive Motion Primitives in Path Planning With Dynamic Obstacles / A. Panov, B. Angulo, K. Yakovlev // IEEE Robotics and Automation Letters. — 2023. — Vol. 8, no. 2. — P. 824—831.
23. *Panov, A.* Task Planning in “Block World” with Deep Reinforcement Learning / A. Panov, E. Ayunts // Biologically Inspired Cognitive Architectures (BICA) for Young Scientists. BICA 2017. Advances in Intelligent Systems and Computing. Vol. 636 / ed. by A. V. Samsonovich, V. V. Klimov. — Springer, 2018. — P. 3—9.
24. *Panov, A.* The Problem of Concept Learning and Goals of Reasoning in Large Language Models / A. Panov, A. A. Chuganskaya, A. K. Kovalev // Hybrid Artificial Intelligent Systems. HAIS 2023. Lecture Notes in Computer Science. Vol. 14001 (HAIS 2023). — Springer, Cham, 2023. — P. 661—672. — (Lecture Notes in Computer Science).
25. *Panov, A.* Self and Other Modelling in Cooperative Resource Gathering with Multi-Agent Reinforcement Learning / A. Panov, V. Davydov, T. Liusko // Brain-Inspired Cognitive Architectures for Artificial Intelligence: BICA*AI 2020. Advances in Intelligent Systems and Computing. Vol. 1310 / ed. by A. V. Samsonovich, R. R. Gudwin, A. d. S. Simões. — Springer International Publishing, 2021. — P. 69—77.
26. *Panov, A.* Applying a Neural Network Architecture with Spatio-Temporal Connections to the Maze Exploration / A. Panov, D. Filin // Biologically Inspired Cognitive Architectures (BICA) for Young Scientists. BICA 2017. Advances in Intelligent Systems and Computing. Vol. 636 / ed. by A. V. Samsonovich, V. V. Klimov. — Springer, 2018. — P. 57—64.

27. *Panov, A.* Learning Adaptive Parking Maneuvers for Self-driving Cars / A. Panov, G. Gorbov, M. Jamal // Proceedings of the Sixth International Scientific Conference “Intelligent Information Technologies for Industry” (IITI’22). IITI 2022. Lecture Notes in Networks and Systems. Vol. 566 (IITI 2022) / ed. by S. Kovalev [et al.]. — 2023. — P. 283—292.
28. *Panov, A.* Model-based Policy Optimization with Neural Differential Equations for Robotic Arm Control / A. Panov, A. Gorodetskiy, K. Mironov // Interactive Collaborative Robotics. ICR 2023. Lecture Notes in Computer Science. Vol. 14214 (ICR 2023). — 2023. — P. 258—266.
29. *Panov, A.* Application of Reinforcement Learning in Open Space Planner for Apollo Auto / A. Panov, D. Ivanov // Proceedings of the Fifth International Scientific Conference “Intelligent Information Technologies for Industry” (IITI’21). IITI 2021. Lecture Notes in Networks and Systems. Vol. 330 / ed. by S. Kovalev [et al.]. — Springer, 2022. — P. 35—43.
30. *Panov, A.* Adaptive Maneuver Planning for Autonomous Vehicles Using Behavior Tree on Apollo Platform / A. Panov, M. Jamal // Artificial Intelligence XXXVIII. SGAI 2021. Lecture Notes in Computer Science. Vol. 13101 / ed. by M. Brammer, R. Ellis. — 2021. — P. 327—340.
31. *Panov, A.* Sign-based Approach to the Task of Role Distribution in the Coalition of Cognitive Agents / A. Panov, G. A. Kiselev // SPIIRAS Proceedings. — 2018. — No. 57. — P. 161—187.
32. *Panov, A.* Hierarchical Psychologically Inspired Planning for Human-Robot Interaction Tasks / A. Panov, G. Kiselev // Interactive Collaborative Robotics. ICR 2019. Lecture Notes in Computer Science. Vol. 11659 / ed. by A. Ronzhin, G. Rigoll, R. Meshcheryakov. — Springer, 2019. — P. 150—160.
33. *Panov, A.* Q-learning of Spatial Actions for Hierarchical Planner of Cognitive Agents / A. Panov, G. Kiselev // Interactive Collaborative Robotics. ICR 2020. Lecture Notes in Computer Science. Vol. 12336 / ed. by A. Ronzhin, G. Rigoll, R. Meshcheryakov. — Springer International Publishing, 2020. — P. 160—169.
34. *Panov, A.* Spatial reasoning and planning in sign-based world model / A. Panov, G. Kiselev, A. Kovalev // Artificial Intelligence. RCAI 2018. Communications in Computer and Information Science. Vol. 934 / ed. by S. Kuznetsov, G. S. Osipov, V. Stefanuk. — Springer, 2018. — P. 1—10.
35. *Panov, A.* Synthesis of the Behavior Plan for Group of Robots with Sign Based World Model / A. Panov, G. A. Kiselev // Interactive Collaborative Robotics. ICR 2017. Lecture Notes in Computer Science. Vol. 10459 / ed. by A. Ronzhin, G. Rigoll, R. Meshcheryakov. — Springer, 2017. — P. 83—94.
36. *Panov, A.* Mental Actions and Modelling of Reasoning in Semiotic Approach to AGI / A. Panov, A. K. Kovalev // Artificial General Intelligence. AGI 2019. Lecture Notes in Computer Science. Vol. 11654 / ed. by P. Hammer [et al.]. — Springer, 2019. — P. 121—131.
37. *Panov, A.* Hyperdimensional Representations in Semiotic Approach to AGI / A. Panov, A. K. Kovalev, E. Osipov // Artificial General Intelligence. AGI 2020. Lecture Notes in Computer Science. Vol. 12177. — Springer, 2020. — P. 231—241.

38. *Panov, A.* Hierarchical Reinforcement Learning with Options and United Neural Network Approximation / A. Panov, V. Kuzmin // Proceedings of the Third International Scientific Conference “Intelligent Information Technologies for Industry” (IITI’18). IITI’18 2018. Advances in Intelligent Systems and Computing. Vol. 874 / ed. by A. Abraham [et al.]. — Springer, 2019. — P. 453—462.
39. *Panov, A.* Navigating Autonomous Vehicle at the Road Intersection Simulator with Reinforcement Learning / A. Panov, M. Martinson, A. Skrynnik // Artificial Intelligence. RAI 2020. Lecture Notes in Computer Science. Vol. 12412 / ed. by S. O. Kuznetsov, A. I. Panov, K. S. Yakovlev. — Springer International Publishing, 2020. — P. 71—84.
40. *Panov, A.* Planning Maneuvers for Autonomous Driving Based on Offline Reinforcement Learning: Comparative Study / A. Panov, M. Melkumov // Proceedings of the Seventh International Scientific Conference “Intelligent Information Technologies for Industry” (IITI’23). IITI 2023. Lecture Notes in Networks and Systems. Vol. 776 (IITI 2023). — Springer, Cham, 2023. — P. 65—74.
41. *Panov, A.* Hierarchical Temporal Memory with Reinforcement Learning / A. Panov, E. Nugamanov // Procedia Computer Science. — 2020. — Vol. 169. — P. 123—131.
42. *Panov, A.* Behavior control as a function of consciousness. I. World model and goal setting / A. Panov, G. S. Osipov, N. V. Chudova // Journal of Computer and Systems Sciences International. — 2014. — Vol. 53, no. 4. — P. 517—529.
43. *Panov, A.* Behavior Control as a Function of Consciousness. II. Synthesis of a Behavior Plan / A. Panov, G. S. Osipov, N. V. Chudova // Journal of Computer and Systems Sciences International. — 2015. — Vol. 54, no. 6. — P. 882—896.
44. *Panov, A.* Relationships and Operations in a Sign-Based World Model of the Actor / A. Panov, G. S. Osipov // Scientific and Technical Information Processing. — 2018. — Vol. 45, no. 5. — P. 317—330.
45. *Panov, A.* Planning Rational Behavior of Cognitive Semiotic Agents in a Dynamic Environment / A. Panov, G. S. Osipov // Scientific and Technical Information Processing. — 2021. — Vol. 48, no. 6. — P. 502—516.
46. *Panov, A.* Flexible Data Augmentation in Off-Policy Reinforcement Learning / A. Panov, A. Rak, A. Skrynnik // Artificial Intelligence and Soft Computing. ICAISC 2021. Lecture Notes in Computer Science. Vol. 12854 / ed. by L. Rutkowski. — Springer, Cham, 2021. — P. 224—235.
47. *Panov, A.* Hierarchical Reinforcement Learning Approach for the Road Intersection Task / A. Panov, M. Shikunov // Biologically Inspired Cognitive Architectures 2019. BICA 2019. Advances in Intelligent Systems and Computing. Vol. 948 / ed. by A. V. Samsonovich. — Springer, 2020. — P. 495—506.
48. *Panov, A.* Hierarchical Reinforcement Learning with Clustering Abstract Machines / A. Panov, A. Skrynnik // Artificial Intelligence. RAI 2019. Communications in Computer and Information Science. Vol. 1093 / ed. by S. O. Kuznetsov, A. I. Panov. — Springer, 2019. — P. 30—43.
49. *Panov, A.* Hierarchical Landmark Policy Optimization for Visual Indoor Navigation / A. Panov, A. Staroverov // IEEE Access. — 2022. — Vol. 10. — P. 70447—70455.

50. *Panov, A.* Hierarchical Actor-Critic with Hindsight for Mobile Robot with Continuous State Space / A. Panov, A. Staroverov // Advances in Neural Computation, Machine Learning, and Cognitive Research III. Studies in Computational Intelligence. Vol. 856 / ed. by B. Kryzhanovsky [et al.]. — Springer, 2020. — P. 62—70.
51. *Panov, A.* Long-Term Exploration in Persistent MDPs / A. Panov, L. Ugadiarov, A. Skrynnik // Advances in Soft Computing. MICAI 2021. Part I. Lecture Notes in Computer Science. Vol. 13067 / ed. by I. Batyrshin, A. Gelbukh, G. Sidorov. — Springer, 2021. — P. 108—120.
52. *Panov, A.* Toward Faster Reinforcement Learning for Robotics : Using Gaussian Processes / A. Panov, A. Younes // RAAI Summer School 2019. Lecture Notes in Computer Science. Vol. 11866 / ed. by G. S. Osipov, A. I. Panov, K. S. Yakovlev. — Springer, 2019. — P. 160—174.
53. *Panov, A.* Sequential Contrastive Learning to Master Effective Representations For Reinforcement Learning and Control / A. Panov, A. Younes // Russian Advances in Artificial Intelligence 2020. RAAI 2020. CEUR Workshop Proceedings. Vol. 2648 / ed. by O. P. Kuznetsov [et al.]. — 2020. — P. 111—121.
54. *Panov, A.* Case-based Task Generalization in Model-based Reinforcement Learning / A. Panov, A. Zholus // Artificial General Intelligence. AGI 2021. Lecture Notes in Computer Science. Vol. 13154 / ed. by B. Goertzel, M. Iklé, A. Potapov. — Springer International Publishing, 2022. — P. 344—354.
55. *Panov, A.* Factorized World Models for Learning Causal Relationships / A. Panov, A. Zholus, Y. Ivchenkov // ICLR Workshop on the Elements of Reasoning: Objects, Structure and Causality. — 2022. — URL: <https://openreview.net/forum?id=BCGfDB0Icec> (visited on May 15, 2024).
56. *Panov, A.* Addressing Task Prioritization in Model-based Reinforcement Learning / A. Panov, A. Zholus, Y. Ivchenkov // Advances in Neural Computation, Machine Learning, and Cognitive Research VI. NEUROINFORMATICS 2022. Vol. 1064 (NeuroInfo 2022) / ed. by B. Kryzhanovsky [et al.]. — Springer, Cham, 2023. — P. 19—30.
57. *Panov, A. I.* Delta Schema Network in Model-based Reinforcement Learning / A. I. Panov, A. Gorodetskiy, A. Shlychkova // Artificial General Intelligence. AGI 2020. Lecture Notes in Computer Science. Vol. 12177 / ed. by B. Goertzel [et al.]. — Springer, 2020. — P. 172—182.
58. *Panov, A. I.* Object-Oriented Decomposition of World Model in Reinforcement Learning / A. I. Panov, L. Ugadiarov // IJCAI Neuro-Symbolic Agents Workshop. — 2023. — URL: <https://nsa-wksp.github.io/assets/papers/Object-Oriented%20Decomposition%20of%20World%20Model%20in%20Reinforcement%20Learning.pdf> (visited on May 15, 2024).
59. *Panov, A. I.* Behavior Planning of Intelligent Agent with Sign World Model / A. I. Panov // Biologically Inspired Cognitive Architectures. — 2017. — Vol. 19. — P. 21—31.
60. *Panov, A. I.* Goal Setting and Behavior Planning for Cognitive Agents / A. I. Panov // Scientific and Technical Information Processing. — 2019. — Vol. 46, no. 6. — P. 404—415.
61. *Panov, A. I.* Simultaneous Learning and Planning in a Hierarchical Control System for a Cognitive Agent / A. I. Panov // Automation and Remote Control. — 2022. — Vol. 83, no. 6. — P. 869—883.
62. *Panov, A. I.* Application of Pretrained Large Language Models in Embodied Artificial Intelligence / A. I. Panov, A. K. Kovalev // Doklady Mathematics. — 2022. — Vol. 106, S1. — S85—S90.

63. *Panov, A. I.* Behavior and Path Planning for the Coalition of Cognitive Robots in Smart Relocation Tasks / A. I. Panov, K. Yakovlev // Robot Intelligence Technology and Applications 4. Advances in Intelligent Systems and Computing. Vol. 447 / ed. by J.-H. Kim [et al.]. — Springer, 2017. — P. 3—20.
64. *Panov, A. I.* Psychologically Inspired Planning Method for Smart Relocation Task / A. I. Panov, K. S. Yakovlev // Procedia Computer Science. Vol. 88. — Elsevier, 2016. — P. 115—124.
65. *Panov, A. I.* Grid Path Planning with Deep Reinforcement Learning: Preliminary Results / A. I. Panov, K. S. Yakovlev, R. Suvorov // Procedia Computer Science. Vol. 123 / ed. by V. Klimov, A. Samsonovich. — Elsevier, 2018. — P. 347—353.
66. *Panov Aleksandr I nad Sarkisyan, C.* Evaluation of Pretrained Large Language Models in Embodied Planning Tasks / C. Panov Aleksandr I nad Sarkisyan, A. K. Kovalev // Artificial General Intelligence. AGI 2023. Lecture Notes in Computer Science. Vol. 13921 (AGI 2023). — Springer Cham, 2023. — P. 222—232.
67. Pathfinding in stochastic environments: learning vs planning / A. Panov [et al.] // PeerJ Computer Science. — 2022. — Vol. 8. — e1056. — URL: <https://peerj.com/articles/cs-1056> (visited on May 15, 2024).
68. Personal Cognitive Assistant: Concept and Key Principals / A. Panov [et al.] // Informatika i ee Primeneniya. — 2019. — Vol. 13, no. 3. — P. 105—113.
69. Personal Cognitive Assistant: Planning Activity with Scripts / A. Panov [et al.] // Informatics and Applications. — 2022. — Vol. 16, no. 1. — P. 46—53.
70. Planning and Learning in Multi-Agent Path Finding / A. Panov [et al.] // Doklady Mathematics. — 2022. — Vol. 106, S1. — S79—S84.
71. Q-Mixing Network for Multi-agent Pathfinding in Partially Observable Grid Environments / A. Panov [et al.] // Artificial Intelligence. RCAI 2021. Lecture Notes in Computer Science. Vol. 12948 / ed. by S. M. Kovalev, S. O. Kuznetsov, A. I. Panov. — Springer, 2021. — P. 169—179. — arXiv: 2108.06148.
72. Quantized Disentangled Representations for Object-Centric Visual Tasks / A. I. Panov [et al.] // Pattern Recognition and Machine Intelligence. PReMI 2023. Lecture Notes in Computer Science. Vol. 14301 (PReMI 2023). — Springer Cham, 2023. — P. 514—522.
73. Question Answering for Visual Navigation in Human-Centered Environments / A. I. Panov [et al.] // Advances in Soft Computing. MICAI 2021. Part II. Lecture Notes in Computer Science. Vol. 13068 / ed. by I. Batyrshin, A. Gelbukh, G. Sidorov. — Springer, 2021. — P. 31—45.
74. Real-Time Object Navigation with Deep Neural Networks and Hierarchical Reinforcement Learning / A. Panov [et al.] // IEEE Access. — 2020. — Vol. 8. — P. 195608—195621.
75. Skill Fusion in Hybrid Robotic Framework for Visual Object Goal Navigation / A. Panov [et al.] // Robotics. — 2023. — Vol. 12. — URL: <https://www.mdpi.com/2218-6581/12/4/104> (visited on May 15, 2024).
76. STRL-Robotics: интеллектуальное управление поведением робототехнической платформы в человеко-ориентированной среде / А. Панов [и др.] // Искусственный интеллект и принятие решений. — 2023. — № 2. — С. 45—63.

77. When to Switch: Planning and Learning For Partially Observable Multi-Agent Pathfinding / A. Panov [et al.] // IEEE Transactions on Neural Networks and Learning Systems. — 2023. — URL: <https://ieeexplore.ieee.org/document/10236574> (visited on May 15, 2024).
78. Знаковая картина мира субъекта поведения / А. И. Панов [и др.]. — М. : Физматлит, 2018. — 264 с.
79. Панов, А. И. Моделирование процесса принятия решения агентом со знаковой картиной мира / А. И. Панов // Теория и практика системного анализа: Труды II Всероссийской научной конференции молодых учёных с международным участием. Т. I. — Рыбинск : РГАТУ имени П.А. Соловьева, 2012. — С. 126—137.
80. Панов, А. И. Семейства отношений в знаковой картине мира / А. И. Панов // Тринадцатая национальная конференция по искусственному интеллекту с международным участием КИИ-2012 (16–20 октября 2012г., г. Белгород, Россия): Труды конференции. Т. 1. — Белгород : Издательство БГТУ, 2012. — С. 301—309.
81. Панов, А. И. Алгебраические свойства операторов распознавания в моделях зрительного восприятия / А. И. Панов // Машинное обучение и анализ данных. — 2014. — Т. 1, № 7. — С. 863—874.
82. Панов, А. И. Представление знаний автономных агентов, планирующих согласованные перемещения / А. И. Панов // Робототехника и техническая кибернетика. — 2015. — № 4. — С. 34—40.
83. Панов, А. И. Представление знаний в задачах согласованного перемещения группы БПЛА / А. И. Панов // Второй Всероссийский научно-практический семинар “Беспилотные транспортные средства с элементами искусственного интеллекта (БТС-ИИ-2015)”, (9 октября 2015г., г. Санкт-Петербург, Россия): Труды семинара. — Санкт-Петербург : Изд-во “Политехника-сервис”, 2015. — С. 74—82.
84. Панов, А. И. Формирование образной компоненты знаний когнитивного агента со знаковой картиной мира / А. И. Панов // Информационные технологии и вычислительные системы. — 2018. — № 4. — С. 84—96.
85. Панов, А. И. STRIPS постановка задачи планирования поведения в знаковой картине мира / А. И. Панов, Г. А. Киселев // Информатика, управление и системный анализ: Труды IV Всероссийской научной конференции молодых учёных с международным участием. Т. I. — Тверь : Тверской государственный технический университет, 2016. — С. 131—138.
86. Панов, А. И. Большие языковые модели как аппроксиматоры значения в знаковой картине мира / А. И. Панов, А. К. Ковалев, А. А. Чуганская // Всероссийская конференция “Поспеловские чтения: искусственный интеллект - проблемы и перспективы Поспеловские чтения-2022 (Москва, 19-20 декабря 2022 г.). Труды конференции. — Издательство ФИЦ ИУ РАН, 2022. — С. 53—70.
87. Панов, А. И. Методы внутренней мотивации в задачах обучения с подкреплением на основе модели / А. И. Панов, А. Латышев // Искусственный интеллект и принятие решений. — 2023. — № 3. — С. 84—97.
88. Панов, А. И. STRL: многоуровневая система управления интеллектуальными агентами / А. И. Панов, Д. А. Макаров, К. С. Яковлев // Пятнадцатая национальная конференция по искусственному интеллекту с международным участием КИИ-2016 (3-7 октября 2016г., г.Смоленск, Россия): Труды конференции. Т. 1. — Смоленск : Универсум, 2016. — С. 179—188.

89. Панов, А. И. Архитектура многоуровневой интеллектуальной системы управления беспилотными летательными аппаратами / А. И. Панов, К. С. Макаров, Д. А. Яковлев // Искусственный интеллект и принятие решений. — 2015. — № 3. — С. 18—33.
90. Панов, А. И. Моделирование поведения автономного мобильного робота / А. И. Панов, А. В. Петров // Вестник РГАТУ имени П.А. Соловьева. — 2012. — № 2. — С. 179—185.
91. Панов, А. И. Когнитивные архитектуры и проекты систем управления автономных мобильных роботов / А. И. Панов, А. В. Петров, Р. Г. Березовский // Вестник РГАТУ имени П.А. Соловьева. — 2013. — № 1. — С. 111—113.
92. Панов, А. И. Автоматическое построение иерархии абстрактных автоматов для задачи обучения с подкреплением / А. И. Панов, А. А. Скрынник // Информатика, управление и системный анализ: Труды V Всероссийской научной конференции молодых учёных с международным участием. — Ростов-на-Дону : Мини-Тайлп, 2018. — С. 7—16.
93. Панов, А. И. Автоматическое формирование правил перемещения с использованием обучения с подкреплением / А. И. Панов, Р. Е. Суворов // Седьмая Международная конференция "Системный анализ и информационные технологии" САИТ-2017 (13-18 июня 2017 г., г. Светлогорск, Россия): Труды конференции. — М. : ФИЦ ИУ РАН, 2017. — С. 303—310.
94. Панов, А. И. Модель среды для актора и критика в обучении с подкреплением / А. И. Панов, Л. Угадяров // Двадцатая Национальная конференция по искусственному интеллекту с международным участием, КИИ-2022 (Москва, 21-23 декабря 2022 г.). Труды конференции. В 2 т. Т. 2. — М. : Издательство МЭИ, 2022. — С. 39—54.
95. Панов, А. И. Взаимодействие стратегического и тактического планирования поведения коалиций агентов в динамической среде / А. И. Панов, К. С. Яковлев // Искусственный интеллект и принятие решений. — 2016. — № 4. — С. 68—78.
96. Панов, А. Иерархическая постановка задачи объектно-центричного обучения с подкреплением / А. Панов // Интегрированные модели и мягкие вычисления в искусственном интеллекте. Сборник научных трудов XI Международной научно-практической конференции (ИММВ-2022, Коломна, 16-19 мая 2022 г.). В 2-х томах. Т. 2. — 2022. — С. 248—256.
97. Панов, А. Формирование умений агента по принципу достижимости в обучении с подкреплением / А. Панов, А. Латышев // Двадцать первая Национальная конференция по искусственному интеллекту с международным участием, КИИ-2023 (Смоленск, 16-20 октября 2023 г.). Труды конференции. В 2 томах. Т. 1 (КИИ-2023). — 2023. — С. 264—274.
98. Принципы построения многоуровневых архитектур систем управления беспилотными летательными аппаратами / А. И. Панов [и др.] // Авиакосмическое приборостроение. — 2013. — № 4. — С. 10—28.

Список литературы

99. A Compositional Object-Based Approach to Learning Physical Dynamics [Электронный ресурс] / M. Chang [et al.] // International Conference on Learning Representations. — 2017. — URL: <https://openreview.net/forum?id=Bkab5dqxe> (visited on May 15, 2024).
100. A computational cognitive framework of spatial memory in brains and robots / T. Madl [et al.] // Cognitive Systems Research. — 2018. — Vol. 47. — P. 147—172.
101. A deep hierarchical approach to lifelong learning in minecraft / C. Tessler [et al.] // Proceedings of the AAAI conference on artificial intelligence. Vol. 31. — 2017. — P. 1553—1561.
102. A differentiable physics engine for deep learning in robotics / J. Degrave [et al.] // Frontiers in neurorobotics. — 2019. — Vol. 13. — P. 6.
103. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play / D. Silver [et al.] // Science. — 2018. — Vol. 362. — P. 1140—1144.
104. A Generalist Agent [Электронный ресурс] / S. Reed [et al.] // Transactions on Machine Learning Research. — 2022. — URL: <https://openreview.net/forum?id=1ikK0kJvj> (visited on May 15, 2024).
105. A generic ROS-based control architecture for pest inspection and treatment in greenhouses using a mobile manipulator / J. Martin [et al.] // IEEE access. — 2021. — Vol. 9. — P. 94981—94995.
106. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity / Y. Bang [et al.] // Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers). — 2023. — P. 675—718.
107. A novel OCR-RCNN for elevator button recognition / D. Zhu [et al.] // 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE. 2018. — P. 3626—3631.
108. A robot arm for pushing elevator buttons / W.-J. Wang [et al.] // Proceedings of SICE Annual Conference 2010. — IEEE. 2010. — P. 1844—1848.
109. A simple neural network module for relational reasoning [Электронный ресурс] / A. Santoro [et al.] // Advances in Neural Information Processing Systems. Vol. 30 / ed. by I. Guyon [et al.]. — Curran Associates, Inc., 2017. — URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/e6acf4b0f69f6f6e60e9a815938aa1ff-Paper.pdf (visited on May 15, 2024).
110. A supportive friend at work: Robotic workplace assistance for the disabled / A. Graser [et al.] // IEEE Robotics & Automation Magazine. — 2013. — Vol. 20, no. 4. — P. 148—159.
111. A Survey of Monte Carlo Tree Search Methods / C. Browne [et al.] // IEEE Transactions on Computational Intelligence and AI in Games. — 2012. — Vol. 4. — P. 1—43.
112. A survey of monte carlo tree search methods / C. B. Browne [et al.] // IEEE Transactions on Computational Intelligence and AI in Games. — 2012. — Vol. 4, no. 1. — P. 1—43.

113. A0C: Alpha Zero in Continuous Action Space [Электронный ресурс] / T. M. Moerland [et al.]. — 2018. — arXiv: 1805.09613 [stat.ML]. — URL: <https://arxiv.org/abs/1805.09613> (visited on May 15, 2024).
114. ACT-R/E: An embodied cognitive architecture for Human-Robot Interaction / G. J. Crafton [et al.] // Journal of Human-Robot Interaction. — 2013. — Vol. 2, no. 1. — P. 30—54.
115. Action-Conditional Video Prediction using Deep Networks in Atari Games [Электронный ресурс] / J. Oh [et al.] // Advances in Neural Information Processing Systems. Vol. 28 / ed. by C. Cortes [et al.]. — Curran Associates, Inc., 2015. — URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/6ba3af5d7b2790e73f0de32e5c8c1798-Paper.pdf (visited on May 15, 2024).
116. Active imitation learning of hierarchical policies / M. Hamidi [et al.] // Proceedings of the 24th International Conference on Artificial Intelligence. — 2015. — P. 3554—3560.
117. Active world model learning with progress curiosity / K. Kim [et al.] // International conference on machine learning. — PMLR. 2020. — P. 5306—5315.
118. Adaptive coverage path planning policy for a cleaning robot with deep reinforcement learning / D. Noh [et al.] // 2022 IEEE International Conference on Consumer Electronics (ICCE). — IEEE. 2022. — P. 1—6.
119. Adaptive Skip Intervals: Temporal Abstraction for Recurrent Dynamical Models [Электронный ресурс] / A. Neitz [et al.] // Advances in Neural Information Processing Systems. Vol. 31 / ed. by S. Bengio [et al.]. — Curran Associates, Inc., 2018. — URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/0f0ee3310223fe38a989b2c818709393-Paper.pdf (visited on May 15, 2024).
120. AgentBench: Evaluating LLMs as Agents [Электронный ресурс] / X. Liu [et al.] // The Twelfth International Conference on Learning Representations. — 2024. — URL: <https://openreview.net/forum?id=zAdUB0aCTQ> (visited on May 15, 2024).
121. AI2-THOR: An Interactive 3D Environment for Visual AI [Электронный ресурс] / E. Kolve [et al.]. — 2022. — arXiv: 1712.05474 [cs.CV]. — URL: <https://arxiv.org/abs/1712.05474> (visited on May 15, 2024).
122. Albus, J. S. 4D/RCS: a reference model architecture for intelligent unmanned ground vehicles / J. S. Albus // Unmanned Ground Vehicle Technology IV. Vol. 4715. — SPIE. 2002. — P. 303—310.
123. Albus, J. S. RCS: A cognitive architecture for intelligent multi-agent systems / J. S. Albus, A. Barbera // Annual Reviews in Control. — 2005. — Vol. 29. — P. 87—99.
124. Alfred: A benchmark for interpreting grounded instructions for everyday tasks / M. Shridhar [et al.] // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. — 2020. — P. 10740—10749.
125. Alvarez, V. M. M. DyNODE: Neural Ordinary Differential Equations for Dynamics Modeling in Continuous Control [Электронный ресурс] / V. M. M. Alvarez, R. Roşca, C. G. Fălcuţescu. — 2020. — arXiv: 2009.04278 [cs.LG]. — URL: <https://arxiv.org/abs/2009.04278> (visited on May 15, 2024).

126. An autonomous eye-in-hand robotic system for elevator button operation based on deep recognition network / D. Zhu [et al.] // IEEE Transactions on Instrumentation and Measurement. — 2020. — Vol. 70. — P. 1—13.
127. An improved frontier-based approach for autonomous exploration / W. Gao [et al.] // 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV). — IEEE. 2018. — P. 292—297.
128. An Independent Evaluation of ChatGPT on MathematicalWord Problems (MWP) [Электронный ресурс] / P. Shakarian [et al.] // Proceedings of the AAAI 2023 Spring Symposium on Challenges Requiring the Combination of Machine Learning and Knowledge Engineering (AAAI-MAKE 2023). — CEUR-WS. 2023. — URL: <https://ceur-ws.org/Vol-3433/paper8.pdf> (visited on May 15, 2024).
129. An Investigation into Pre-Training Object-Centric Representations for Reinforcement Learning / J. Yoon [et al.] // International Conference on Machine Learning. — PMLR. 2023. — P. 40147—40174.
130. An Investigation into the Open World Survival Game Crafter [Электронный ресурс] / A. Stanić [et al.] // Decision Awareness in Reinforcement Learning Workshop at ICML 2022. — 2022. — URL: <https://openreview.net/forum?id=C5q00n9dBxC> (visited on May 15, 2024).
131. *Anderson, J. R.* ACT-R: A Theory of Higher Level Cognition and Its Relation to Visual Attention / J. R. Anderson, M. Matessa, C. Lebiere // Human–Computer Interaction. — 1997. — Vol. 12. — P. 439—462.
132. *Andre, D.* Generalized Prioritized Sweeping [Электронный ресурс] / D. Andre, N. Friedman, R. Parr // Advances in Neural Information Processing Systems. Vol. 10 / ed. by M. Jordan, M. Kearns, S. Solla. — MIT Press, 1997. — URL: https://proceedings.neurips.cc/paper_files/paper/1997/file/7b5b23f4aadf9513306bcd59afb6e4c9-Paper.pdf (visited on May 15, 2024).
133. *Anthony, T.* Thinking Fast and Slow with Deep Learning and Tree Search [Электронный ресурс] / T. Anthony, Z. Tian, D. Barber // Advances in Neural Information Processing Systems. Vol. 30 / ed. by I. Guyon [et al.]. — Curran Associates, Inc., 2017. — URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/d8e1344e27a5b08cdfd5d027d9b8d6de-Paper.pdf (visited on May 15, 2024).
134. *Arkin, R. C.* Motor schema based navigation for a mobile robot: An approach to programming by behavior / R. C. Arkin // Proceedings. 1987 IEEE International Conference on Robotics and Automation. Vol. 4. — 1987. — P. 264—271.
135. Asynchronous methods for deep reinforcement learning / V. Mnih [et al.] // International conference on machine learning. — PMLR. 2016. — P. 1928—1937.
136. Attend, Infer, Repeat: Fast Scene Understanding with Generative Models [Электронный ресурс] / S. M. A. Eslami [et al.] // Advances in Neural Information Processing Systems. Vol. 29 / ed. by D. Lee [et al.]. — Curran Associates, Inc., 2016. — URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/52947e0ade57a09e4a1386d08f17b656-Paper.pdf (visited on May 15, 2024).

137. Attention is All you Need [Электронный ресурс] / A. Vaswani [et al.] // Advances in Neural Information Processing Systems. Vol. 30 / ed. by I. Guyon [et al.]. — Curran Associates, Inc., 2017. — URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fdb053c1c4a845aa-Paper.pdf (visited on May 15, 2024).
138. Aubret, A. A survey on intrinsic motivation in reinforcement learning [Электронный ресурс] / A. Aubret, L. Matignon, S. Hassas. — 2019. — arXiv: 1908.06976 [cs.LG]. — URL: <https://arxiv.org/abs/1908.06976> (visited on May 15, 2024).
139. Aubret, A. An information-theoretic perspective on intrinsic motivation in reinforcement learning: A survey / A. Aubret, L. Matignon, S. Hassas // Entropy. — 2023. — Vol. 25, no. 2. — P. 327.
140. Auer, P. Finite-time analysis of the multiarmed bandit problem / P. Auer, N. Cesa-Bianchi, P. Fischer // Machine learning. — 2002. — Vol. 47. — P. 235—256.
141. Auxiliary tasks and exploration enable objectgoal navigation / J. Ye [et al.] // Proceedings of the IEEE/CVF international conference on computer vision. — 2021. — P. 16117—16126.
142. Ba, J. L. Layer Normalization [Электронный ресурс] / J. L. Ba, J. R. Kiros, G. E. Hinton. — 2016. — arXiv: 1607.06450 [stat.ML]. — URL: <https://arxiv.org/abs/1607.06450> (visited on May 15, 2024).
143. Bacon, P.-L. The Option-Critic Architecture / P.-L. Bacon, J. Harb, D. Precup // Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17). — 2017. — P. 1726—1734.
144. Bara, C.-P. MindCraft: Theory of Mind Modeling for Situated Dialogue in Collaborative Tasks / C.-P. Bara, C.-W. Sky, J. Chai // Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. — 2021. — P. 1112—1125.
145. Barto, A. G. Intrinsic motivation for reinforcement learning systems / A. G. Barto, O. Simsek // Proceedings of the thirteenth yale workshop on adaptive and learning systems. — Yale University Press. 2005. — P. 113—118.
146. Barto, A. G. Recent Advances in Hierarchical Reinforcement Learning / A. G. Barto, S. Mahadevan // Discrete Event Dynamic Systems. — 2003. — Vol. 13. — P. 341—379.
147. Bayesian controller fusion: Leveraging control priors in deep reinforcement learning for robotics / K. Rana [et al.] // The International Journal of Robotics Research. — 2023. — Vol. 42, no. 3. — P. 123—146.
148. Belkin, I. Real-time lidar-based localization of mobile ground robot / I. Belkin, A. Abramenko, D. Yudin // Procedia Computer Science. — 2021. — Vol. 186. — P. 440—448.
149. Bellman, R. E. A Markovian decision processes / R. E. Bellman // Journal of Mathematics and Mechanics. — 1957. — Vol. 6, no. 5. — P. 679—684.
150. Bellman, R. E. Applied Dynamic Programming / R. E. Bellman, S. E. Dreyfus ; The RAND Corporation. — London, 1962. — 384 p.
151. BenchBot: Evaluating Robotics Research in Photorealistic 3D Simulation and on Real Robots [Электронный ресурс] / B. Talbot [et al.]. — 2020. — arXiv: 2008.00635 [cs.RO]. — URL: <https://arxiv.org/abs/2008.00635> (visited on May 15, 2024).

152. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin [et al.] // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). — 2019. — P. 4171—4186.
153. *Besold, T. R.* Towards integrated neural-symbolic systems for human-level AI: Two research programs helping to bridge the gaps / T. R. Besold, K. U. Kuhnberger // Biologically Inspired Cognitive Architectures. — 2015. — Vol. 14. — P. 97—110.
154. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework [Электронный ресурс] / I. Higgins [et al.] // International Conference on Learning Representations. — 2017. — URL: <https://openreview.net/forum?id=Sy2fzU9gl> (visited on May 15, 2024).
155. Binding Actions to Objects in World Models [Электронный ресурс] / O. Biza [et al.] // ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality. — 2022. — URL: <https://openreview.net/forum?id=HImz8BuUclc> (visited on May 15, 2024).
156. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model [Электронный ресурс] / T. L. Scao [et al.]. — 2023. — arXiv: 2211.05100 [cs.CL]. — URL: <https://arxiv.org/abs/2211.05100> (visited on May 15, 2024).
157. *Blum, A. L.* Fast planning through planning graph analysis / A. L. Blum, M. L. Furst // Artificial intelligence. — 1997. — Vol. 90, no. 1/2. — P. 281—300.
158. Bootstrapping from Game Tree Search [Электронный ресурс] / J. Veness [et al.] // Advances in Neural Information Processing Systems. Vol. 22 / ed. by Y. Bengio [et al.]. — Curran Associates, Inc., 2009. — URL: https://proceedings.neurips.cc/paper_files/paper/2009/file/389bc7bb1e1c2a5e7e147703232a88f6-Paper.pdf (visited on May 15, 2024).
159. *Borrajo, D.* Progress in Case-Based Planning / D. Borrajo, A. Roubíčková, I. Serina // ACM Computing Surveys. — 2015. — Vol. 47, no. 2. — P. 1—39.
160. *Bothell, D.* ACT-R 7 Reference Manual / D. Bothell ; Carnegie Mellon University. — 2015. — 516 p.
161. Bridging the Gap to Real-World Object-Centric Learning [Электронный ресурс] / M. Seitzer [et al.] // The Eleventh International Conference on Learning Representations. — 2023. — URL: https://openreview.net/forum?id=b9tUk-f_ag (visited on May 15, 2024).
162. *Brunskill, E.* Pac-inspired option discovery in lifelong reinforcement learning / E. Brunskill, L. Li // International conference on machine learning. — PMLR. 2014. — P. 316—324.
163. Builder, we have done it: evaluating & extending dialogue-AMR NLU pipeline for two collaborative domains / C. Bonial [et al.] // Proceedings of the 14th International Conference on Computational Semantics (IWCS). — 2021. — P. 173—183.
164. *Buskey, G. D.* Autonomous helicopter hover using an artificial neural network / G. D. Buskey, G. F. Wyeth, J. M. Roberts // . Vol. 2. — 2001. — P. 1635—1640.
165. Byte Pair encoding: A text compression scheme that accelerates pattern matching / Y. Shibata [et al.]. — 1999. — 13 p.

166. Can ChatGPT Understand Too? A Comparative Study on ChatGPT and Fine-tuned BERT [Электронный ресурс] / Q. Zhong [et al.]. — 2023. — arXiv: 2302.10198 [cs.CL]. — URL: <https://arxiv.org/abs/2302.10198> (visited on May 15, 2024).
167. Cardoso, R. C. Decentralised Planning for Multi-Agent Programming Platforms / R. C. Cardoso, R. H. Bordini // Adaptive Agents and Multi-Agent Systems. — 2019. — P. 799—807.
168. CaRINA Intelligent Robotic Car: Architectural design and applications / L. C. Fernandes [et al.] // Journal of Systems Architecture. — 2014. — Vol. 60. — P. 372—392.
169. Carpenter, G. A. The ART of adaptive pattern recognition by a self-organizing neural network / G. A. Carpenter, S. Grossberg // Computer. — 1987. — Vol. 21. — P. 77—88.
170. Cattell, R. B. Handbook for the sixteen personality factor questionnaire (16 PF) / R. B. Cattell, H. W. Eber, M. M. Tatsuoka. — 1992.
171. CausalWorld: A Robotic Manipulation Benchmark for Causal Structure and Transfer Learning [Электронный ресурс] / O. Ahmed [et al.] // International Conference on Learning Representations. — 2021. — URL: <https://openreview.net/forum?id=SK7A5pdrgov> (visited on May 15, 2024).
172. CC-News-En: A large English news corpus / J. Mackenzie [et al.] // Proceedings of the 29th ACM International Conference on Information & Knowledge Management. — 2020. — P. 3077—3084.
173. Chain-of-thought prompting elicits reasoning in large language models / J. Wei [et al.] // Advances in Neural Information Processing Systems. Vol. 35. — 2022. — P. 24824—24837.
174. Chang, M. Object representations as fixed points: Training iterative refinement algorithms with implicit differentiation / M. Chang, T. Griffiths, S. Levine // Advances in Neural Information Processing Systems. Vol. 35. — 2022. — P. 32694—32708.
175. Choi, D. Evolution of the ICARUS cognitive architecture / D. Choi, P. Langley // Cognitive Systems Research. — 2018. — Vol. 48. — P. 25—38.
176. Christodoulou, P. Soft Actor-Critic for Discrete Action Settings [Электронный ресурс] / P. Christodoulou. — 2019. — arXiv: 1910.07207 [cs.LG]. — URL: <https://arxiv.org/abs/1910.07207> (visited on May 15, 2024).
177. Çimen, T. State-Dependent Riccati Equation (SDRE) Control: A Survey / T. Çimen // IFAC Proceedings Volumes. — 2008. — Vol. 41. — P. 3761—3775.
178. Classification and recall with binary hyperdimensional computing: Tradeoffs in choice of density and mapping characteristics / D. Kleyko [et al.] // IEEE transactions on neural networks and learning systems. — 2018. — Vol. 29, no. 12. — P. 5880—5898.
179. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning / J. Johnson [et al.] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2017. — P. 2901—2910.
180. CLEVRER: Collision Events for Video Representation and Reasoning [Электронный ресурс] / K. Yi* [et al.] // International Conference on Learning Representations. — 2020. — URL: <https://openreview.net/forum?id=HkxYzANYDB> (visited on May 15, 2024).
181. Clone-structured graph representations enable flexible learning and vicarious evaluation of cognitive maps / D. George [et al.] // Nature Communications. — 2021. — Vol. 12, no. 1. — P. 2392.

182. *Cloutier, J. R.* The state-dependent nonlinear regulator with state constraints / J. R. Cloutier, J. C. Cockburn // Proceedings of the 2001 American Control Conference. Vol. 1. — 2001. — P. 390—395.
183. *Cloutier, J. R.* Dynamic conversion of flight path angle commands to body attitude commands / J. R. Cloutier, D. T. Stansbery // Proceedings of the 2002 American Control Conference. Vol. 1. — 2002. — P. 221—225.
184. COBRA: Data-Efficient Model-Based RL through Unsupervised Object Discovery and Curiosity-Driven Exploration [Электронный ресурс] / N. Watters [et al.]. — 2019. — arXiv: 1905.09275 [cs.LG]. — URL: <https://arxiv.org/abs/1905.09275> (visited on May 15, 2024).
185. Code as policies: Language model programs for embodied control / J. Liang [et al.] // 2023 IEEE International Conference on Robotics and Automation (ICRA). — IEEE. 2023. — P. 9493—9500.
186. Cognitive mapping and planning for visual navigation / S. Gupta [et al.] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2017. — P. 2616—2625.
187. Collaborative autonomy for manned/unmanned teams / S. Jameson [et al.] // Annual Forum Proceedings-American Helicopter Society. Vol. 61. — Citeseer. 2005. — P. 1673.
188. Combined reinforcement learning via abstract representations / V. François-Lavet [et al.] // Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. — 2019. — P. 3582—3589.
189. Combined reinforcement learning via abstract representations / V. François-Lavet [et al.] // Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. — 2019. — P. 3582—3589.
190. Combining Q-Learning and Search with Amortized Value Estimates [Электронный ресурс] / J. B. Hamrick [et al.] // International Conference on Learning Representations. — 2020. — URL: <https://openreview.net/forum?id=SkeAaJrKDS> (visited on May 15, 2024).
191. Combining Safe Interval Path Planning and Constrained Path Following Control: Preliminary Results / K. Yakovlev [et al.] // Interactive Collaborative Robotics. ICR 2019. Lecture Notes in Computer Science. — 2019. — P. 310—319.
192. Conditional Object-Centric Learning from Video [Электронный ресурс] / T. Kipf [et al.] // International Conference on Learning Representations. — 2022. — URL: https://openreview.net/forum?id=aD7uesX1GF_ (visited on May 15, 2024).
193. Contact force control of an aerial manipulator in pressing an emergency switch process / X. Meng [et al.] // 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE. 2018. — P. 2107—2113.
194. Contextual imagined goals for self-supervised robotic learning / A. Nair [et al.] // Conference on Robot Learning. — PMLR. 2020. — P. 530—539.
195. Contingency-Aware Exploration in Reinforcement Learning [Электронный ресурс] / J. Choi [et al.] // International Conference on Learning Representations. — 2019. — URL: <https://openreview.net/forum?id=HyxGB2AcY7> (visited on May 15, 2024).
196. Continual model-based reinforcement learning with hypernetworks / Y. Huang [et al.] // 2021 IEEE International Conference on Robotics and Automation (ICRA). — IEEE. 2021. — P. 799—805.

197. Continuous control with deep reinforcement learning [Электронный ресурс] / T. P. Lillicrap [et al.]. — 2019. — arXiv: 1509.02971 [cs.LG]. — URL: <https://arxiv.org/abs/1509.02971> (visited on May 15, 2024).
198. Continuous deep q-learning with model-based acceleration / S. Gu [et al.] // International conference on machine learning. — PMLR. 2016. — P. 2829—2838.
199. Corneil, D. Efficient model-based deep reinforcement learning with variational state tabulation / D. Corneil, W. Gerstner, J. Brea // International Conference on Machine Learning. — PMLR. 2018. — P. 1049—1058.
200. Curiosity-driven Exploration by Self-supervised Prediction / D. Pathak [et al.] // Proceedings of the 34th International Conference on Machine Learning. — 2017. — P. 2778—2787.
201. Curiosity-driven exploration by self-supervised prediction / D. Pathak [et al.] // International conference on machine learning. — PMLR. 2017. — P. 2778—2787.
202. Cut and learn for unsupervised object detection and instance segmentation / X. Wang [et al.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2023. — P. 3124—3134.
203. Daniel, T. Unsupervised Image Representation Learning with Deep Latent Particles / T. Daniel, A. Tamar // International Conference on Machine Learning. — PMLR. 2022. — P. 4644—4665.
204. Data-Efficient Hierarchical Reinforcement Learning [Электронный ресурс] / O. Nachum [et al.] // Advances in Neural Information Processing Systems. Vol. 31 / ed. by S. Bengio [et al.]. — Curran Associates, Inc., 2018. — URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/e6384711491713d29bc63fc5eeb5ba4f-Paper.pdf (visited on May 15, 2024).
205. Dayan, P. Feudal Reinforcement Learning / P. Dayan, G. E. Hinton. — 1992.
206. DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames [Электронный ресурс] / E. Wijmans [et al.] // International Conference on Learning Representations. — 2020. — URL: <https://openreview.net/forum?id=H1gX8C4YPr> (visited on May 15, 2024).
207. Dean, T. L. Planning and Control / T. L. Dean, M. P. Wellman. — San Mateo, California : Morgan Kaufmann, 1991. — 486 p.
208. Dearden, R. Model based Bayesian exploration / R. Dearden, N. Friedman, D. Andre // Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence. — 1999. — P. 150—159.
209. Dearden, R. Bayesian Q-learning / R. Dearden, N. Friedman, S. Russell // Proceedings of the 15th National Conference on Artificial Intelligence. — 1998. — P. 761—768.
210. Dec-MCTS: Decentralized planning for multi-robot active perception / G. Best [et al.] // The International Journal of Robotics Research. — 2019. — Vol. 38, no. 2/3. — P. 316—337.
211. Deep Exploration via Bootstrapped DQN [Электронный ресурс] / I. Osband [et al.] // Advances in Neural Information Processing Systems. Vol. 29 / ed. by D. Lee [et al.]. — Curran Associates, Inc., 2016. — URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/8d8818c8e140c64c743113f563cf750f-Paper.pdf (visited on May 15, 2024).
212. Deep hierarchical planning from pixels / D. Hafner [et al.] // Advances in Neural Information Processing Systems. Vol. 35. — 2022. — P. 26091—26104.

213. Deep intrinsically motivated continuous actor-critic for efficient robotic visuomotor skill learning / M. B. Hafez [et al.] // Paladyn, Journal of Behavioral Robotics. — 2019. — Vol. 10, no. 1. — P. 14—29.
214. Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning [Электронный ресурс] / X. Guo [et al.] // Advances in Neural Information Processing Systems. Vol. 27 / ed. by Z. Ghahramani [et al.]. — Curran Associates, Inc., 2014. — URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/8bb88f80d334b1869781beb89f7b73be-Paper.pdf (visited on May 15, 2024).
215. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates / S. Gu [et al.] // 2017 IEEE international conference on robotics and automation (ICRA). — IEEE. 2017. — P. 3389—3396.
216. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models [Электронный ресурс] / K. Chua [et al.] // Advances in Neural Information Processing Systems. Vol. 31 / ed. by S. Bengio [et al.]. — Curran Associates, Inc., 2018. — URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/3de568f8597b94bda53149c7d7f5958c-Paper.pdf (visited on May 15, 2024).
217. Deep reinforcement learning using genetic algorithm for parameter optimization / A. Sehgal [et al.] // 2019 Third IEEE International Conference on Robotic Computing (IRC). — IEEE. 2019. — P. 596—601.
218. Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data [Электронный ресурс] / M. Karl [et al.] // International Conference on Learning Representations. — 2017. — URL: <https://openreview.net/forum?id=HyTqHL5xg> (visited on May 15, 2024).
219. DeepProbLog: Neural Probabilistic Logic Programming [Электронный ресурс] / R. Manhaeve [et al.] // Advances in Neural Information Processing Systems. Vol. 31 / ed. by S. Bengio [et al.]. — Curran Associates, Inc., 2018. — URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/dc5d637ed5e62c36ecb73b654b05ba2a-Paper.pdf (visited on May 15, 2024).
220. Deisenroth, M. PILCO: A model-based and data-efficient approach to policy search / M. Deisenroth, C. E. Rasmussen // Proceedings of the 28th International Conference on machine learning (ICML-11). — 2011. — P. 465—472.
221. Dennett, D. C. Beliefs about beliefs / D. C. Dennett // Behavioral and Brain sciences. — 1978. — Vol. 1, no. 4. — P. 568—570.
222. Denoising criterion for variational auto-encoding framework / D. Im Im [et al.] // Proceedings of the AAAI conference on artificial intelligence. Vol. 31. — 2017. — P. 2059—2065.
223. Design of an intelligent outdoor mobile robot with autonomous road-crossing function for urban environments / A. N. Chand [et al.] // 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM). — IEEE. 2012. — P. 355—362.
224. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost / H. Zhu [et al.] // 2019 International Conference on Robotics and Automation (ICRA). — IEEE. 2019. — P. 3651—3657.

225. Dialfred: Dialogue-enabled agents for embodied instruction following / X. Gao [et al.] // IEEE Robotics and Automation Letters. — 2022. — Vol. 7, no. 4. — P. 10049—10056.
226. Direct Preference Optimization: Your Language Model is Secretly a Reward Model / R. Rafailov [et al.] // Advances in Neural Information Processing Systems. Vol. 36 / ed. by A. Oh [et al.]. — Curran Associates, Inc., 2023. — P. 53728—53741.
227. Discovering and achieving goals via world models / R. Mendonca [et al.] // Advances in Neural Information Processing Systems. Vol. 34. — 2021. — P. 24379—24391.
228. Discovering faster matrix multiplication algorithms with reinforcement learning / A. Fawzi [et al.] // Nature. — 2022. — Vol. 610, no. 7930. — P. 47—53.
229. Disentangling the independently controllable factors of variation by interacting with the world [Электронный ресурс] / V. Thomas [et al.]. — 2018. — arXiv: 1802.09484 [stat.ML]. — URL: <https://arxiv.org/abs/1802.09484> (visited on May 15, 2024).
230. *Diuk, C.* An object-oriented representation for efficient reinforcement learning / C. Diuk, A. Cohen, M. L. Littman // Proceedings of the 25th International Conference on Machine Learning. — 2008. — P. 240—247.
231. Diversity is All You Need: Learning Skills without a Reward Function [Электронный ресурс] / B. Eysenbach [et al.] // International Conference on Learning Representations. — 2019. — URL: <https://openreview.net/forum?id=SJx63jRqFm> (visited on May 15, 2024).
232. dm_control: Software and tasks for continuous control / S. Tunyasuvunakool [et al.] // Software Impacts. — 2020. — Vol. 6. — P. 100022.
233. Do as i can, not as i say: Grounding language in robotic affordances / A. Brohan [et al.] // Conference on Robot Learning. — PMLR. 2023. — P. 287—318.
234. *Donadello, I.* Logic tensor networks for semantic image interpretation / I. Donadello, L. Serafini, A. D. Garcez // Proceedings of the 26th International Joint Conference on Artificial Intelligence. — 2017. — P. 1596—1602.
235. *Dorri, A.* Multi-Agent Systems: A Survey / A. Dorri, S. S. Kanhere, R. Jurdak // IEEE Access. — 2018. — Vol. 6. — P. 28573—28593.
236. Dota 2 with Large Scale Deep Reinforcement Learning [Электронный ресурс] / C. Berner [et al.]. — 2019. — arXiv: 1912.06680 [cs.LG]. — URL: <https://arxiv.org/abs/1912.06680> (visited on May 15, 2024).
237. Dream to Control: Learning Behaviors by Latent Imagination [Электронный ресурс] / D. Hafner [et al.] // International Conference on Learning Representations. — 2020. — URL: <https://openreview.net/forum?id=S110TC4tDS> (visited on May 15, 2024).
238. *Du, J.* Model-based reinforcement learning for semi-markov decision processes with neural odes / J. Du, J. Futoma, F. Doshi-Velez // Advances in Neural Information Processing Systems. Vol. 33. — 2020. — P. 19805—19816.
239. Dueling network architectures for deep reinforcement learning / Z. Wang [et al.] // International conference on machine learning. — PMLR. 2016. — P. 1995—2003.

240. Dyna-style planning with linear function approximation and prioritized sweeping / R. S. Sutton [et al.] // Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence. — 2008. — P. 528—536.
241. Dynamic abstraction in reinforcement learning via clustering / S. Mannor [et al.] // Proceedings of the twenty-first international conference on Machine learning. — 2004. — P. 71.
242. *Dzhivelikian, E.* Learning Hidden Markov Model of Stochastic Environment with Bio-Inspired Probabilistic Temporal Memory / E. Dzhivelikian, P. Kudrov, A. I. Panov // Biologically Inspired Cognitive Architectures 2023. Studies in Computational Intelligence. Vol. 1130 (BICA 2023). — 2024. — P. 330—339.
243. *Efroni, Y.* Online Planning with Lookahead Policies / Y. Efroni, M. Ghavamzadeh, S. Mannor // Advances in Neural Information Processing Systems. Vol. 33 / ed. by H. Larochelle [et al.]. — Curran Associates, Inc., 2020. — P. 14024—14033.
244. *Elfes, A.* Using occupancy grids for mobile robot perception and navigation / A. Elfes // Computer. — 1989. — Vol. 22. — P. 46—57.
245. Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images / J. Watter [et al.] // Advances in neural information processing systems. — 2015. — P. 2746—2754.
246. EMI: Exploration with Mutual Information / H. Kim [et al.] // International Conference on Machine Learning. — PMLR. 2019. — P. 3360—3369.
247. End-to-End Differentiable Physics for Learning and Control [Электронный ресурс] / F. de Avila Belbute-Peres [et al.] // Advances in Neural Information Processing Systems. Vol. 31 / ed. by S. Bengio [et al.]. — Curran Associates, Inc., 2018. — URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/842424a1d0595b76ec4fa03c46e8d755-Paper.pdf (visited on May 15, 2024).
248. End-to-end neural speaker diarization with permutation-free objectives / Y. Fujita [et al.] // Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. Vol. 2019. — 2019. — P. 4300—4304.
249. End-to-end object detection with transformers / N. Carion [et al.] // European conference on computer vision. — Springer. 2020. — P. 213—229.
250. End-to-end robotic reinforcement learning without reward engineering [Электронный ресурс] / A. Singh [et al.] // Robotics: Science and Systems 2019. — 2019. — P. 3389—3396. — (Visited on May 15, 2024).
251. *Engel, J.* LSD-SLAM: Large-scale direct monocular SLAM / J. Engel, T. Schöps, D. Cremers // European conference on computer vision. — Springer. 2014. — P. 834—849.
252. *Engelcke, M.* Genesis-v2: Inferring unordered object representations without iterative refinement / M. Engelcke, O. Parker Jones, I. Posner // Advances in Neural Information Processing Systems. Vol. 34. — 2021. — P. 8085—8094.
253. Episodic Curiosity through Reachability [Электронный ресурс] / N. Savinov [et al.] // International Conference on Learning Representations. — 2019. — URL: <https://openreview.net/forum?id=SkeK3s0qKQ> (visited on May 15, 2024).

254. Evaluating Large Language Models Trained on Code [Электронный ресурс] / M. Chen [et al.]. — 2021. — arXiv: 2107.03374 [cs.LG]. — URL: <https://arxiv.org/abs/2107.03374> (visited on May 15, 2024).
255. Exploration by random network distillation [Электронный ресурс] / Y. Burda [et al.] // International Conference on Learning Representations. — 2019. — URL: <https://openreview.net/forum?id=H11JJnR5Ym> (visited on May 15, 2024).
256. Exploring the limits of transfer learning with a unified text-to-text transformer / C. Raffel [et al.] // Journal of machine learning research. — 2020. — Vol. 21, no. 140. — P. 1—67.
257. Facmac: Factored multi-agent centralised policy gradients / B. Peng [et al.] // Advances in Neural Information Processing Systems. Vol. 34. — 2021. — P. 12208—12221.
258. Fast Exploration with Simplified Models and Approximately Optimistic Planning in Model Based Reinforcement Learning [Электронный ресурс] / R. Keramati [et al.]. — 2018. — arXiv: 1806.00175 [cs.AI]. — URL: <https://arxiv.org/abs/1806.00175> (visited on May 15, 2024).
259. FeUdal Networks for Hierarchical Reinforcement Learning / A. S. Vezhnevets [et al.] // Proceedings of the 34th International Conference on Machine Learning / ed. by D. Precup, Y. W. Teh. — PMLR, 2017. — P. 3540—3549.
260. *Fierro, R.* Control of a nonholomic mobile robot: Backstepping kinematics into dynamics / R. Fierro, F. L. Lewis // Journal of robotic systems. — 1997. — Vol. 14, no. 3. — P. 149—163.
261. *Filippis, L. de.* Path Planning Strategies for UAVS in 3D Environments / L. de Filippis, G. Guglieri, F. Quagliotti // Journal of Intelligent & Robotic Systems. — 2011. — Vol. 65. — P. 247—264.
262. FILM: Following Instructions in Language with Modular Methods [Электронный ресурс] / S. Y. Min [et al.] // International Conference on Learning Representations. — 2022. — URL: <https://openreview.net/forum?id=qI4542Y2s1D> (visited on May 15, 2024).
263. Finetuned Language Models are Zero-Shot Learners [Электронный ресурс] / J. Wei [et al.] // International Conference on Learning Representations. — 2022. — URL: <https://openreview.net/forum?id=gEZrGCozdqR> (visited on May 15, 2024).
264. *Finn, C.* Unsupervised Learning for Physical Interaction through Video Prediction [Электронный ресурс] / C. Finn, I. Goodfellow, S. Levine // Advances in Neural Information Processing Systems. Vol. 29 / ed. by D. Lee [et al.]. — Curran Associates, Inc., 2016. — URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/d9d4f495e875a2e075a1a4a6e1b9770f-Paper.pdf (visited on May 15, 2024).
265. *Florensa, C.* Stochastic Neural Networks for Hierarchical Reinforcement Learning [Электронный ресурс] / C. Florensa, Y. Duan, P. Abbeel // International Conference on Learning Representations. — 2017. — URL: <https://openreview.net/forum?id=B1oK8aoxe> (visited on May 15, 2024).
266. *Fox, M.* PDDL2.1: An extension to PDDL for expressing temporal planning domains / M. Fox, D. Long // Journal of Artificial Intelligence Research. — 2003. — Vol. 20. — P. 61—124.
267. *Fugal, J.* On the impact of gravity compensation on reinforcement learning in goal-reaching tasks for robotic manipulators / J. Fugal, J. Bae, H. A. Poonawala // Robotics. — 2021. — Vol. 10, no. 1. — P. 46.

268. *Gadzicki, K.* Early vs late fusion in multimodal convolutional neural networks / K. Gadzicki, R. Khamsehashari, C. Zetzsche // 2020 IEEE 23rd international conference on information fusion (FUSION). — IEEE. 2020. — P. 1—6.
269. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium [Электронный ресурс] / M. Heusel [et al.] // Advances in Neural Information Processing Systems. Vol. 30 / ed. by I. Guyon [et al.]. — Curran Associates, Inc., 2017. — URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefef65871369074926d-Paper.pdf (visited on May 15, 2024).
270. *Garcez, A. d.* Neurosymbolic AI: The 3 rd wave / A. d. Garcez, L. C. Lamb // Artificial Intelligence Review. — 2023. — Vol. 56, no. 11. — P. 12387—12406.
271. *Garcia, R.* The Implementation of an Autonomous Helicopter Testbed / R. Garcia, K. P. Valavanis // Journal of Intelligent and Robotic Systems. — 2009. — Vol. 54. — P. 423—454.
272. *Gat, E.* Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots [Электронный ресурс] / E. Gat // AAAI Conference on Artificial Intelligence. — 1992. — URL: https://www.freelug.eu/IMG/pdf/Integrating_Planning_and_Reacting_in_a_Heterogeneous_Asynchronous_Architecture.pdf (visited on May 15, 2024).
273. *Gat, E.* Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots / E. Gat // Proceedings of the tenth national conference on Artificial intelligence. Vol. 1992. — 1992. — P. 809—815.
274. *Ge, H.* An Open-IoT Approach on Elevator for Enabling Autonomous Robotic Vertical Mobility / H. Ge, M. Matsui, N. Koshizuka // 2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech). — IEEE. 2021. — P. 139—141.
275. Generalizing plans to new environments in relational MDPs / C. Guestrin [et al.] // Proceedings of the 18th international joint conference on Artificial intelligence. — 2003. — P. 1003—1010.
276. Generating Instructions at Different Levels of Abstraction / A. Köhn [et al.] // Proceedings of the 28th International Conference on Computational Linguistics. — 2020. — P. 2802—2813.
277. Generative pretraining from pixels / M. Chen [et al.] // International conference on machine learning. — PMLR. 2020. — P. 1691—1703.
278. GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations [Электронный ресурс] / M. Engelcke [et al.] // International Conference on Learning Representations. — 2020. — URL: <https://openreview.net/forum?id=BkxfaTVFwH> (visited on May 15, 2024).
279. *George, D.* How the brain might work: A hierarchical and temporal model for learning and recognition / D. George. — Stanford University, 2008. — 177 p.
280. *George, D.* Towards a mathematical theory of cortical micro-circuits / D. George, J. Hawkins // PLoS computational biology. — 2009. — Vol. 5, no. 10. — e1000532.
281. *Gerevini, A. E.* Plan Constraints and Preferences in PDDL3 / A. E. Gerevini, D. Long ; Department of Electronics for Automation, University of Brescia. — 2005. — 12 p.

282. *Ghallab, M.* Automated Planning Theory and Practice / M. Ghallab, D. Nau, P. Traverso. — Morgan Kaufmann, 2004. — 638 p.
283. *Ghallab, M.* Automated Planning and Acting / M. Ghallab, D. Nau, P. Traverso. — 2016. — 354 p.
284. *Ghidini, C.* Distributed first order logic / C. Ghidini, L. Serafini // Artificial Intelligence. — 2017. — Vol. 253. — P. 1—39.
285. *Ghosh, D.* Learning Actionable Representations with Goal Conditioned Policies [Электронный ресурс] / D. Ghosh, A. Gupta, S. Levine // International Conference on Learning Representations. — 2019. — URL: <https://openreview.net/forum?id=Hye9lnCct7> (visited on May 15, 2024).
286. *Ghosh, D.* Learning Actionable Representations with Goal Conditioned Policies [Электронный ресурс] / D. Ghosh, A. Gupta, S. Levine // International Conference on Learning Representations. — 2019. — URL: <https://openreview.net/forum?id=Hye9lnCct7> (visited on May 15, 2024).
287. Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning / B. Riviere [et al.] // IEEE Robotics and Automation Letters. — 2020. — Vol. 5, no. 3. — P. 4249—4256.
288. Globally consistent 3D mapping with scan matching / D. Borrmann [et al.] // Robotics and Autonomous Systems. — 2008. — Vol. 56. — P. 130—142.
289. Goel, S. Subgoal discovery for hierarchical reinforcement learning using learned policies / S. Goel, M. Huber // The Florida AI Research Society. — 2003. — P. 346—350.
290. Goertzel, B. From Abstract Agents Models to Real-World AGI Architectures: Bridging the Gap / B. Goertzel // Lecture Notes in Computer Science. Vol. 10414 / ed. by T. Everitt, B. Goertzel, A. Potapov. — Springer International Publishing, 2017. — P. 3—12.
291. GPT-4 Technical Report [Электронный ресурс] / OpenAI [et al.]. — 2024. — arXiv: 2303.08774 [cs.CL]. — URL: <https://arxiv.org/abs/2303.08774> (visited on May 15, 2024).
292. GPT-NeoX-20B: An Open-Source Autoregressive Language Model / S. Black [et al.] // Proceedings of BigScience Episode# 5—Workshop on Challenges & Perspectives in Creating Large Language Models. — 2022. — P. 95—136.
293. Gramopadhye, M. Generating executable action plans with environmentally-aware language models / M. Gramopadhye, D. Szafir // 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE. 2023. — P. 3568—3575.
294. Grandmaster level in StarCraft II using multi-agent reinforcement learning / O. Vinyals [et al.] // Nature. — 2019. — Vol. 575. — P. 350—354.
295. Graph neural networks for decentralized multi-robot path planning / Q. Li [et al.] // Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2020). — IEEE. 2020. — P. 11785—11792.
296. Graves, A. Neural Turing Machines [Электронный ресурс] / A. Graves, G. Wayne, I. Danihelka. — 2014. — arXiv: 1410.5401 [cs.NE]. — URL: <https://arxiv.org/abs/1410.5401> (visited on May 15, 2024).

297. *Greff, K.* Neural Expectation Maximization [Электронный ресурс] / K. Greff, S. van Steenkiste, J. Schmidhuber // Advances in Neural Information Processing Systems. Vol. 30 / ed. by I. Guyon [et al.]. — Curran Associates, Inc., 2017. — URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/d2cd33e9c0236a8c2d8bd3fa91ad3acf-Paper.pdf (visited on May 15, 2024).
298. *Greff, K.* On the Binding Problem in Artificial Neural Networks [Электронный ресурс] / K. Greff, S. van Steenkiste, J. Schmidhuber. — 2020. — arXiv: 2012.05208 [cs.NE]. — URL: <https://arxiv.org/abs/2012.05208> (visited on May 15, 2024).
299. *Gregor, K.* Variational Intrinsic Control [Электронный ресурс] / K. Gregor, D. J. Rezende, D. Wierstra. — 2016. — arXiv: 1611.07507 [cs.LG]. — URL: <https://arxiv.org/abs/1611.07507> (visited on May 15, 2024).
300. *Greydanus, S.* Hamiltonian Neural Networks [Электронный ресурс] / S. Greydanus, M. Dzamba, J. Yosinski // Advances in Neural Information Processing Systems. Vol. 32 / ed. by H. Wallach [et al.]. — Curran Associates, Inc., 2019. — URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/26cd8ecadce0d4efd6cc8a8725cbd1f8-Paper.pdf (visited on May 15, 2024).
301. *Grossberg, S.* Towards solving the hard problem of consciousness: The varieties of brain resonances and the conscious experiences that they support / S. Grossberg // Neural Networks. — 2017. — Vol. 87. — P. 38—95.
302. *Gudwin, R.* Semiotics and intelligent systems development / R. Gudwin, J. Queiroz. — 2006. — 351 p.
303. *Ha, D.* Recurrent World Models Facilitate Policy Evolution [Электронный ресурс] / D. Ha, J. Schmidhuber // Advances in Neural Information Processing Systems. Vol. 31 / ed. by S. Bengio [et al.]. — Curran Associates, Inc., 2018. — URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf (visited on May 15, 2024).
304. *Ha, D.* World models [Электронный ресурс] / D. Ha, J. Schmidhuber. — 2018. — arXiv: 1803.101221 [cs.LG]. — URL: <https://arxiv.org/abs/1803.10122> (visited on May 15, 2024).
305. Habitat 2.0: Training home assistants to rearrange their habitat / A. Szot [et al.] // Advances in neural information processing systems. Vol. 34. — 2021. — P. 251—266.
306. Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI [Электронный ресурс] / S. K. Ramakrishnan [et al.] // Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2). — 2021. — URL: <https://openreview.net/forum?id=-v40uqNs5P> (visited on May 15, 2024).
307. Habitat: A platform for embodied ai research / M. Savva [et al.] // Proceedings of the IEEE/CVF international conference on computer vision. — 2019. — P. 9339—9347.
308. *Hafner, D.* Benchmarking the Spectrum of Agent Capabilities [Электронный ресурс] / D. Hafner // International Conference on Learning Representations. — 2022. — URL: <https://openreview.net/forum?id=1W0z96MFEoH> (visited on May 15, 2024).
309. *Harabor, D.* Online graph pruning for pathfinding on grid maps / D. Harabor, A. Grastien // Proceedings of the AAAI conference on artificial intelligence. Vol. 25. — 2011. — P. 1114—1119.

310. *Harnad, S.* Symbol Grounding Problem / S. Harnad // Physica. — 1990. — Vol. 42. — P. 335—346.
311. *Hart, P. E.* A formal basis for the heuristic determination of minimum cost paths / P. E. Hart, N. J. Nilsson, B. Raphael // IEEE transactions on Systems Science and Cybernetics. — 1968. — Vol. 4, no. 2. — P. 100—107.
312. *Hasselt, H. van.* Deep Reinforcement Learning with Double Q-learning / H. van Hasselt, A. Guez, D. Silver // Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 30 (AAAI). — 2016. — P. 2094—2100.
313. *Hawkins, J.* A Theory of How Columns in the Neocortex Enable Learning the Structure of the World / J. Hawkins, S. Ahmad, Y. Cui // Frontiers in Neural Circuits. — 2017. — Vol. 11. — P. 295079.
314. *Hélie, S.* Autonomous learning in psychologically-oriented cognitive architectures: A survey / S. Hélie, R. Sun // New Ideas in Psychology. — 2014. — Vol. 34, no. 1. — P. 37—55.
315. *Hengst, B.* Hierarchical Approaches / B. Hengst // Reinforcement Learning. — 2012. — P. 293—323.
316. *Hester, T.* Generalized model learning for reinforcement learning on a humanoid robot / T. Hester, M. Quinlan, P. Stone // 2010 IEEE International Conference on Robotics and Automation. — IEEE. 2010. — P. 2369—2374.
317. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation [Электронный ресурс] / T. D. Kulkarni [et al.] // Advances in Neural Information Processing Systems. Vol. 29 / ed. by D. Lee [et al.]. — Curran Associates, Inc., 2016. — URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/f442d33fa06832082290ad8544a8da27-Paper.pdf (visited on May 15, 2024).
318. Hierarchical Planning: Relating Task and Goal Decomposition with Task Sharing / R. Alford [et al.] // Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16). — 2016. — P. 3022—3029.
319. *Hochreiter, S.* Long short-term memory / S. Hochreiter, J. Schmidhuber // Neural computation. — 1997. — Vol. 9, no. 8. — P. 1735—1780.
320. *Holland, G. Z.* The Effect of Planning Shape on Dyna-style Planning in High-dimensional State Spaces [Электронный ресурс] / G. Z. Holland, E. J. Talvitie, M. Bowling. — 2019. — arXiv: 1806.01825 [cs.AI]. — URL: <https://arxiv.org/abs/1806.01825> (visited on May 15, 2024).
321. Human-guided Collaborative Problem Solving: A Natural Language based Framework [Электронный ресурс] / H. Kokel [et al.]. — 2022. — arXiv: 2207.09566 [cs.HC]. — URL: <https://arxiv.org/abs/2207.09566> (visited on May 15, 2024).
322. Human-level Atari 200x faster [Электронный ресурс] / S. Kapturowski [et al.] // The Eleventh International Conference on Learning Representations. — 2023. — URL: <https://openreview.net/forum?id=JtC6y0HRoJJ> (visited on May 15, 2024).
323. Human-level control through deep reinforcement learning / V. Mnih [et al.] // Nature. — 2015. — Vol. 518, no. 7540. — P. 529—533.

324. Hypertree proof search for neural theorem proving / G. Lample [et al.] // Advances in Neural Information Processing Systems. Vol. 35. — 2022. — P. 26337—26349.
325. IGLU Gridworld: Simple and Fast Environment for Embodied Dialog Agents [Электронный ресурс] / A. Zholus [et al.] // CVPR Workshop on Embodied AI. — 2022. — URL: <http://arxiv.org/abs/2206.00142> (visited on May 15, 2024).
326. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures / L. Espeholt [et al.] // International conference on machine learning. — PMLR. 2018. — P. 1407—1416.
327. Implementation of nonlinear model predictive path-following control for an industrial robot / T. Faulwasser [et al.] // IEEE Transactions on Control Systems Technology. — 2016. — Vol. 25, no. 4. — P. 1505—1511.
328. Inner Monologue: Embodied Reasoning through Planning with Language Models / W. Huang [et al.] // Conference on Robot Learning. — PMLR. 2023. — P. 1769—1782.
329. Intelligent mobile robot controller design for hotel room service with deep learning arm-based elevator manipulator / P.-Y. Yang [et al.] // 2018 International Conference on System Science and Engineering (ICSSE). — IEEE. 2018. — P. 1—6.
330. Interaction Networks for Learning about Objects, Relations and Physics [Электронный ресурс] / P. Battaglia [et al.] // Advances in Neural Information Processing Systems. Vol. 29 / ed. by D. Lee [et al.]. — Curran Associates, Inc., 2016. — URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/3147da8ab4a0437c15ef51a5cc7f2dc4-Paper.pdf (visited on May 15, 2024).
331. Interactive Grounded Language Understanding in a Collaborative Environment: IGLU 2021 / J. Kiseleva [et al.] // Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track, PMLR. Vol. 176. — 2022. — P. 146—161.
332. Interactive language: Talking to robots in real time [Электронный ресурс] / C. Lynch [et al.] // IEEE Robotics and Automation Letters. — 2023. — URL: <https://ieeexplore.ieee.org/abstract/document/10182264> (visited on May 15, 2024).
333. Intrinsically motivated learning in natural and artificial systems / G. Baldassarre, M. Mirolli, [et al.]. — Springer, 2013. — 465 p.
334. Invariant Slot Attention: Object Discovery with Slot-Centric Reference Frames / O. Biza [et al.] // International Conference on Machine Learning. — PMLR. 2023. — P. 2507—2527.
335. Is Curiosity All You Need? On the Utility of Emergent Behaviours from Curious Exploration [Электронный ресурс] / O. Groth [et al.]. — 2021. — arXiv: 2109.08603 [cs.LG]. — URL: <https://arxiv.org/abs/2109.08603> (visited on May 15, 2024).
336. *Jang, E.* Categorical Reparameterization with Gumbel-Softmax [Электронный ресурс] / E. Jang, S. Gu, B. Poole // International Conference on Learning Representations. — 2017. — URL: <https://openreview.net/forum?id=rkE3y85ee> (visited on May 15, 2024).
337. *Jayannavar, P.* Learning to execute instructions in a Minecraft dialogue / P. Jayannavar, A. Narayan-Chen, J. Hockenmaier // Proceedings of the 58th annual meeting of the association for computational linguistics. — 2020. — P. 2589—2602.

338. *Jiang, D.* Feedback-based tree search for reinforcement learning / D. Jiang, E. Ekwedike, H. Liu // International conference on machine learning. — PMLR. 2018. — P. 2284—2293.
339. *Jonathan, B.* TDLeaf (λ): Combining temporal difference learning with game-tree search / B. Jonathan // Proc. of the 9th Australian Conference on Neural Networks (ACNN-98). — 1998. — P. 39—43.
340. *Jong, N. K.* Model-based function approximation in reinforcement learning / N. K. Jong, P. Stone // Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems. — 2007. — P. 1—8.
341. *Jordan, M. I.* Forward Models: Supervised Learning with a Distal Teacher / M. I. Jordan, D. E. Rumelhart // Cognitive Science. — 1992. — Vol. 16. — P. 307—354.
342. *Ju, C.* Path planning using an improved a-star algorithm / C. Ju, Q. Luo, X. Yan // 2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan). — IEEE. 2020. — P. 23—26.
343. *Kaelbling, L. P.* Planning and acting in partially observable stochastic domains / L. P. Kaelbling, M. L. Littman, A. R. Cassandra // Artificial Intelligence. — 1998. — Vol. 101, no. 1. — P. 99—134.
344. *Kaelbling, L. P.* Integrated task and motion planning in belief space / L. P. Kaelbling, T. Lozano-Perez // The International Journal of Robotics Research. — 2013. — Vol. 32. — P. 1194—1227.
345. *Kakade, S.* Approximately optimal approximate reinforcement learning / S. Kakade, J. Langford // Proceedings of the Nineteenth International Conference on Machine Learning. — 2002. — P. 267—274.
346. *Kalweit, G.* Uncertainty-driven imagination for continuous deep reinforcement learning / G. Kalweit, J. Boedecker // Conference on robot learning. — PMLR. 2017. — P. 195—206.
347. *Kamthe, S.* Data-efficient reinforcement learning with probabilistic model predictive control / S. Kamthe, M. Deisenroth // International conference on artificial intelligence and statistics. — PMLR. 2018. — P. 1701—1710.
348. *Karazija, L.* ClevrTex: A Texture-Rich Benchmark for Unsupervised Multi-Object Segmentation [Электронный ресурс] / L. Karazija, I. Laina, C. Rupprecht // Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2). — 2021. — URL: <https://openreview.net/forum?id=J4N12qRMDrR> (visited on May 15, 2024).
349. *Kendoul, F.* Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems / F. Kendoul // Journal of Field Robotics. — 2012. — Vol. 29, no. 2. — P. 315—378.
350. *Kim, H.* Disentangling by factorising / H. Kim, A. Mnih // International conference on machine learning. — PMLR. 2018. — P. 2649—2658.
351. *Kingma, D. P.* Auto-Encoding Variational Bayes [Электронный ресурс] / D. P. Kingma, M. Welling. — 2022. — arXiv: 1312.6114 [stat.ML]. — URL: <https://arxiv.org/abs/1312.6114> (visited on May 15, 2024).
352. *Kingma, D. P.* Adam: A Method for Stochastic Optimization [Электронный ресурс] / D. P. Kingma, J. Ba. — 2017. — arXiv: 1412.6980 [cs.LG]. — URL: <https://arxiv.org/abs/1412.6980> (visited on May 15, 2024).

353. *Kipf, T.* Contrastive Learning of Structured World Models [Электронный ресурс] / T. Kipf, E. van der Pol, M. Welling // International Conference on Learning Representations. — 2020. — URL: <https://openreview.net/forum?id=H1gax6VtDB> (visited on May 15, 2024).
354. *Kipf, T.* Contrastive Learning of Structured World Models [Электронный ресурс] / T. Kipf, E. van der Pol, M. Welling // International Conference on Learning Representations. — 2020. — URL: <https://openreview.net/forum?id=H1gax6VtDB> (visited on May 15, 2024).
355. *Klein, G. S. W.* Parallel Tracking and Mapping for Small AR Workspaces / G. S. W. Klein, D. W. Murray // 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality. — 2007. — P. 225—234.
356. *Klissarov, M.* Flexible option learning / M. Klissarov, D. Precup // Advances in Neural Information Processing Systems. — 2021. — Vol. 34. — P. 4632—4646.
357. *Klyubin, A. S.* All else being equal be empowered / A. S. Klyubin, D. Polani, C. L. Nehaniv // European Conference on Artificial Life. — Springer. 2005. — P. 744—753.
358. *Kochenderfer, M. J.* Decision Making Under Uncertainty: Theory and Application / M. J. Kochenderfer. — MIT Press, 2015. — 322 p.
359. *Kocsis, L.* Bandit based monte-carlo planning / L. Kocsis, C. Szepesvári // Machine Learning: ECML 2006: 17th European Conference on Machine Learning Berlin, Germany, September 18-22, 2006 Proceedings 17. — Springer. 2006. — P. 282—293.
360. *Koenig, S.* Lifelong planning A* / S. Koenig, M. Likhachev, D. Furcy // Artificial Intelligence. — 2004. — Vol. 155, no. 1/2. — P. 93—146.
361. *Kojima, N.* To Learn or Not to Learn: Analyzing the Role of Learning for Navigation in Virtual Environments [Электронный ресурс] / N. Kojima, J. Deng. — 2019. — arXiv: 1907.11770 [cs.CV]. — URL: <https://arxiv.org/abs/1907.11770> (visited on May 15, 2024).
362. *Kosinski, M.* Evaluating Large Language Models in Theory of Mind Tasks [Электронный ресурс] / M. Kosinski. — 2024. — arXiv: 2302.02083 [cs.CL]. — URL: <https://arxiv.org/abs/2302.02083> (visited on May 15, 2024).
363. *Kosiorek, A. R.* Conditional Set Generation with Transformers [Электронный ресурс] / A. R. Kosiorek, H. Kim, D. J. Rezende. — 2020. — arXiv: 2006.16841 [cs.CV]. — URL: <https://arxiv.org/abs/2006.16841> (visited on May 15, 2024).
364. *Kothari, M.* A Probabilistically Robust Path Planning Algorithm for UAVs Using Rapidly-Exploring Random Trees / M. Kothari, I. Postlethwaite // Journal of Intelligent & Robotic Systems. — 2013. — Vol. 71. — P. 231—253.
365. *Krishnan, R. G.* Deep Kalman Filters [Электронный ресурс] / R. G. Krishnan, U. Shalit, D. Sontag. — 2015. — arXiv: 1511.05121 [stat.ML]. — URL: <https://arxiv.org/abs/1511.05121> (visited on May 15, 2024).
366. *Ktistakis, I. P.* Assistive intelligent robotic wheelchairs / I. P. Ktistakis, N. G. Bourbakis // IEEE Potentials. — 2017. — Vol. 36, no. 1. — P. 10—13.

367. *Kuffner, J. J.* RRT-connect: An efficient approach to single-query path planning / J. J. Kuffner, S. M. LaValle // Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). Vol. 2. — IEEE. 2000. — P. 995—1001.
368. *Kuhn, H. W.* The Hungarian method for the assignment problem / H. W. Kuhn // Naval research logistics quarterly. — 1955. — Vol. 2, no. 1/2. — P. 83—97.
369. *Labbé, M.* RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation / M. Labbé, F. Michaud // Journal of field robotics. — 2019. — Vol. 36, no. 2. — P. 416—446.
370. *Lai, M.* Giraffe: Using Deep Reinforcement Learning to Play Chess [Электронный ресурс] / M. Lai. — 2015. — arXiv: 1509.01549 [cs.AI]. — URL: <https://arxiv.org/abs/1509.01549> (visited on May 15, 2024).
371. *Laird, J. E.* The Soar Cognitive Architecture / J. E. Laird. — MIT Press, 2012. — 374 p.
372. *Laird, J. E.* SOAR: An Architecture for General Intelligence / J. E. Laird, A. Newell, P. S. Rosenbloom // Artificial intelligence. — 1987. — Vol. 33. — P. 1—64.
373. LaMDA: Language Models for Dialog Applications [Электронный ресурс] / R. Thoppilan [et al.]. — 2022. — arXiv: 2201.08239 [cs.CL]. — URL: <https://arxiv.org/abs/2201.08239> (visited on May 15, 2024).
374. *Lange, S.* Batch reinforcement learning / S. Lange, T. Gabel, M. Riedmiller // Reinforcement learning: State-of-the-art. — Springer, 2012. — P. 45—73.
375. Language and culture internalization for human-like autotelic AI / C. Colas [et al.] // Nature Machine Intelligence. — 2022. — Vol. 4, no. 12. — P. 1068—1076.
376. Language models are few-shot learners / T. Brown [et al.] // Advances in neural information processing systems. Vol. 33. — 2020. — P. 1877—1901.
377. Language models are unsupervised multitask learners / A. Radford [et al.] // OpenAI blog. — 2019. — Vol. 1, no. 8. — P. 9.
378. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents / W. Huang [et al.] // International Conference on Machine Learning. — PMLR. 2022. — P. 9118—9147.
379. Lara – Human-guided collaborative problem solver: Effective integration of learning, reasoning and communication [Электронный ресурс] / H. Kokel [et al.] // Tenth Annual Conference on Advances in Cognitive Systems (ACS). — 2022. — URL: https://harshakokel.com/pdf/Kokel_ACS2022.pdf (visited on May 15, 2024).
380. Large Batch Simulation for Deep Reinforcement Learning [Электронный ресурс] / B. Shacklett [et al.] // International Conference on Learning Representations. — 2021. — URL: <https://openreview.net/forum?id=cP5IcoAkfKa> (visited on May 15, 2024).
381. Learning an Embedding Space for Transferable Robot Skills [Электронный ресурс] / K. Hausman [et al.] // International Conference on Learning Representations. — 2018. — URL: <https://openreview.net/forum?id=rk07ZXZRb> (visited on May 15, 2024).

382. Learning and Planning with a Semantic Model [Электронный ресурс] / Y. Wu [et al.]. — 2018. — arXiv: 1809.10842 [cs.LG]. — URL: <https://arxiv.org/abs/1809.10842> (visited on May 15, 2024).
383. Learning Continuous Control Policies by Stochastic Value Gradients [Электронный ресурс] / N. Heess [et al.] // Advances in Neural Information Processing Systems. Vol. 28 / ed. by C. Cortes [et al.]. — Curran Associates, Inc., 2015. — URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/148510031349642de5ca0c544f31b2ef-Paper.pdf (visited on May 15, 2024).
384. Learning Dynamics Model in Reinforcement Learning by Incorporating the Long Term Future [Электронный ресурс] / N. R. Ke [et al.]. — 2019. — arXiv: 1903.01599 [stat.ML]. — URL: <https://arxiv.org/abs/1903.01599> (visited on May 15, 2024).
385. Learning Invariant Representations for Reinforcement Learning without Reconstruction [Электронный ресурс] / A. Zhang [et al.] // International Conference on Learning Representations. — 2021. — URL: <https://openreview.net/forum?id=-2FCwDKRREu> (visited on May 15, 2024).
386. Learning latent dynamics for planning from pixels / D. Hafner [et al.] // 36th International Conference on Machine Learning. — 2019. — P. 4528—4547.
387. Learning model-based planning from scratch [Электронный ресурс] / R. Pascanu [et al.]. — 2017. — arXiv: 1707.06170 [cs.AI]. — URL: <https://arxiv.org/abs/1707.06170> (visited on May 15, 2024).
388. Learning Multi-Level Hierarchies with Hindsight [Электронный ресурс] / A. Levy [et al.]. — 2019. — arXiv: 1712.00948 [cs.AI]. — URL: <https://arxiv.org/abs/1712.00948> (visited on May 15, 2024).
389. Learning navigation behaviors end-to-end with autorl / H.-T. L. Chiang [et al.] // IEEE Robotics and Automation Letters. — 2019. — Vol. 4, no. 2. — P. 2007—2014.
390. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation / K. Cho [et al.] // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). — 2014. — P. 1724—1734.
391. Learning Plannable Representations with Causal InfoGAN [Электронный ресурс] / T. Kurutach [et al.] // Advances in Neural Information Processing Systems. Vol. 31 / ed. by S. Bengio [et al.]. — Curran Associates, Inc., 2018. — URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/08aac6ac98e59e523995c161e57875f5-Paper.pdf (visited on May 15, 2024).
392. Learning Representations and Generative Models for 3D Point Clouds [Электронный ресурс] / P. Achlioptas [et al.]. — 2018. — arXiv: 1707.02392 [cs.CV]. — URL: <https://arxiv.org/abs/1707.02392> (visited on May 15, 2024).
393. Learning To Explore Using Active Neural SLAM [Электронный ресурс] / D. S. Chaplot [et al.] // International Conference on Learning Representations. — 2020. — URL: <https://openreview.net/forum?id=Hk1Xn1BKDH> (visited on May 15, 2024).
394. Learning to Generalize With Object-Centric Agents in the Open World Survival Game Crafter / A. Stanić [et al.] // IEEE Transactions on Games. — 2024. — Vol. 16, no. 2. — P. 384—395.

395. Learning to Guide and to be Guided in the Architect-Builder Problem [Электронный ресурс] / P. Barde [et al.] // International Conference on Learning Representations. — 2022. — URL: <https://openreview.net/forum?id=swiyAeGzFhQ> (visited on May 15, 2024).
396. Learning to guide task and motion planning using score-space representation / B. Kim [et al.] // International Journal of Robotics Research. — 2019. — Vol. 38, no. 7. — P. 793—812.
397. Learning to Model the World with Language [Электронный ресурс] / J. Lin [et al.]. — 2024. — arXiv: 2308.01399 [cs.CL]. — URL: <https://arxiv.org/abs/2308.01399> (visited on May 15, 2024).
398. Learning to Poke by Poking: Experiential Learning of Intuitive Physics [Электронный ресурс] / P. Agrawal [et al.] // Advances in Neural Information Processing Systems. Vol. 29 / ed. by D. Lee [et al.]. — Curran Associates, Inc., 2016. — URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/c203d8a151612acf12457e4d67635a95-Paper.pdf (visited on May 15, 2024).
399. Learning to summarize with human feedback / N. Stiennon [et al.] // Advances in Neural Information Processing Systems. Vol. 33 / ed. by H. Larochelle [et al.]. — Curran Associates, Inc., 2020. — P. 3008—3021.
400. Learning transferable visual models from natural language supervision / A. Radford [et al.] // International conference on machine learning. — PMLR. 2021. — P. 8748—8763.
401. Learning Visual Predictive Models of Physics for Playing Billiards [Электронный ресурс] / K. Fragkiadaki [et al.]. — 2016. — arXiv: 1511.07404 [cs.CV]. — URL: <https://arxiv.org/abs/1511.07404> (visited on May 15, 2024).
402. LEBP – Language Expectation & Binding Policy: A Two-Stream Framework for Embodied Vision-and-Language Interaction Task Learning Agents [Электронный ресурс] / H. Liu [et al.]. — 2022. — arXiv: 2203.04637 [cs.AI]. — URL: <https://arxiv.org/abs/2203.04637> (visited on May 15, 2024).
403. Levine, S. Learning Neural Network Policies with Guided Policy Search under Unknown Dynamics [Электронный ресурс] / S. Levine, P. Abbeel // Advances in Neural Information Processing Systems. Vol. 27 / ed. by Z. Ghahramani [et al.]. — Curran Associates, Inc., 2014. — URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/6766aa2750c19aad2fa1b32f36ed4aee-Paper.pdf (visited on May 15, 2024).
404. Levine, S. Guided policy search / S. Levine, V. Koltun // International conference on machine learning. — PMLR. 2013. — P. 1—9.
405. Lgsvl simulator: A high fidelity simulator for autonomous driving / G. Rong [et al.] // 2020 IEEE 23rd International conference on intelligent transportation systems (ITSC). — IEEE. 2020. — P. 1—6.
406. Li, N. Learning object-centric representations of multi-object scenes from multiple views / N. Li, C. Eastwood, R. Fisher // Advances in Neural Information Processing Systems. Vol. 33. — 2020. — P. 5656—5666.
407. Lieto, A. A computational framework for concept representation in cognitive systems and architectures: Concepts as heterogeneous proxytypes / A. Lieto // Procedia Computer Science. — 2014. — Vol. 41. — P. 6—14.

408. Lifelong multi-agent path finding in large-scale warehouses / J. Li [et al.] // Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021). — 2021. — P. 11272—11281.
409. *Likhachev, M.* R* search / M. Likhachev, A. Stentz // Proceedings of the 23rd national conference on Artificial intelligence-Volume 1. — 2008. — P. 344—350.
410. *Liu, J.* Recognizing elevator buttons and labels for blind navigation / J. Liu, Y. Tian // 2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER). — IEEE. 2017. — P. 1236—1240.
411. Llm-planner: Few-shot grounded planning for embodied agents with large language models / C. H. Song [et al.] // Proceedings of the IEEE/CVF International Conference on Computer Vision. — 2023. — P. 2998—3009.
412. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action / D. Shah, B. Osiński, S. Levine, [et al.] // Conference on Robot Learning. — PMLR. 2023. — P. 492—504.
413. *Luong, M.-T.* Effective Approaches to Attention-based Neural Machine Translation / M.-T. Luong, H. Pham, C. D. Manning // Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. — 2015. — P. 1412—1421.
414. *Ma, Z.* Distributed heuristic multi-agent path finding with communication / Z. Ma, Y. Luo, H. Ma // 2021 IEEE International Conference on Robotics and Automation (ICRA). — IEEE. 2021. — P. 8699—8705.
415. *Machado, M. C.* A laplacian framework for option discovery in reinforcement learning / M. C. Machado, M. G. Bellemare, M. Bowling // International Conference on Machine Learning. — PMLR. 2017. — P. 2295—2304.
416. *Mahadevan, S.* Learning Representation and Control in Markov Decision Processes: New Frontiers / S. Mahadevan // Foundations and Trends in Machine Learning. — 2009. — Vol. 1. — P. 403—565.
417. *Mann, T.* Scaling up approximate value iteration with options: Better policies with fewer iterations / T. Mann, S. Mannor // International conference on machine learning. — PMLR. 2014. — P. 127—135.
418. Mastering Atari Games with Limited Data / W. Ye [et al.] // Advances in Neural Information Processing Systems. Vol. 34 / ed. by M. Ranzato [et al.]. — Curran Associates, Inc., 2021. — P. 25476—25488.
419. Mastering Atari with Discrete World Models [Электронный ресурс] / D. Hafner [et al.] // International Conference on Learning Representations. — 2021. — URL: <https://openreview.net/forum?id=0oabwyZb0u> (visited on May 15, 2024).
420. Mastering Atari, Go, chess and shogi by planning with a learned model / J. Schrittwieser [et al.] // Nature. — 2019. — Vol. 588. — P. 604—609.
421. Mastering Diverse Domains through World Models [Электронный ресурс] / D. Hafner [et al.]. — 2024. — arXiv: 2301.04104 [cs.AI]. — URL: <https://arxiv.org/abs/2301.04104> (visited on May 15, 2024).
422. Mastering the game of Go with deep neural networks and tree search / D. Silver [et al.] // Nature. — 2016. — Vol. 529, no. 7587. — P. 484—489.

423. Mastering the game of Go without human knowledge / D. Silver [et al.] // Nature. — 2017. — Vol. 550. — P. 354—359.
424. Maximum a posteriori policy optimisation [Электронный ресурс] / A. Abdolmaleki [et al.] // 6th International Conference on Learning Representations. — 2018. — URL: <https://openreview.net/forum?id=S1ANxQW0b> (visited on May 15, 2024).
425. *McAllister, R. T.* Improving PILCO with Bayesian Neural Network Dynamics Models / R. T. McAllister, C. E. Rasmussen // Data-efficient machine learning workshop, ICML. Vol. 4. — 2016. — P. 25.
426. *McGovern, A.* Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density / A. McGovern, A. G. Barto // Proceedings of the 18th International Conference on Machine Learning, 2001. — 2001. — P. 361—368.
427. Mean Actor Critic [Электронный ресурс] / C. Allen [et al.]. — 2018. — arXiv: 1709.00503 [stat.ML]. — URL: <https://arxiv.org/abs/1709.00503> (visited on May 15, 2024).
428. *Menache, I.* Q-Cut—Dynamic Discovery of Sub-goals in Reinforcement Learning / I. Menache, S. Mannor, N. Shimkin // ECML 2002: Machine Learning: ECML 2002. — 2002. — P. 295—306.
429. Meta Learning Shared Hierarchies [Электронный ресурс] / K. Frans [et al.] // International Conference on Learning Representations. — 2018. — URL: <https://openreview.net/forum?id=SyXO1eWAW> (visited on May 15, 2024).
430. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning / T. Yu [et al.] // Conference on robot learning. — PMLR. 2020. — P. 1094—1100.
431. Metacontrol for Adaptive Imagination-Based Optimization [Электронный ресурс] / J. B. Hamrick [et al.] // International Conference on Learning Representations. — 2017. — URL: <https://openreview.net/forum?id=Bk8BvDqex> (visited on May 15, 2024).
432. Microsoft coco: Common objects in context / T.-Y. Lin [et al.] // Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13. — Springer. 2014. — P. 740—755.
433. *Migimatsu, T.* Object-Centric Task and Motion Planning in Dynamic Environments / T. Migimatsu, J. Bohg // IEEE Robotics and Automation Letters. — 2020. — Vol. 5. — P. 844—851.
434. Minedojo: Building open-ended embodied agents with internet-scale knowledge / L. Fan [et al.] // Advances in Neural Information Processing Systems. Vol. 35. — 2022. — P. 18343—18362.
435. *Minsky, M. L.* Frame-system theory / M. L. Minsky // Thinking. — 1977. — P. 355—376.
436. *Mishra, N.* Prediction and control with temporal segment models / N. Mishra, P. Abbeel, I. Mordatch // International conference on machine learning. — PMLR. 2017. — P. 2459—2468.
437. Mobile manipulation hackathon: Moving into real world applications / M. A. Roa [et al.] // IEEE Robotics & Automation Magazine. — 2021. — Vol. 28, no. 2. — P. 112—124.
438. Mobile robot path planning in dynamic environments through globally guided reinforcement learning / B. Wang [et al.] // IEEE Robotics and Automation Letters. — 2020. — Vol. 5, no. 4. — P. 6932—6939.

439. Model Based Reinforcement Learning for Atari [Электронный ресурс] / Ł. Kaiser [et al.] // International Conference on Learning Representations. — 2020. — URL: <https://openreview.net/forum?id=S1xCPJHtDB> (visited on May 15, 2024).
440. Model predictive path following control for autonomous cars considering a measurable disturbance: Implementation, testing, and verification / H. Guo [et al.] // Mechanical Systems and Signal Processing. — 2019. — Vol. 118. — P. 41—60.
441. Model-based reinforcement learning with value-targeted regression / A. Ayoub [et al.] // International Conference on Machine Learning. — PMLR. 2020. — P. 463—474.
442. Modeling Parts, Structure, and System Dynamics via Predictive Learning [Электронный ресурс] / Z. Liu [et al.] // International Conference on Learning Representations. — 2019. — URL: <https://openreview.net/forum?id=rJe10iC5K7> (visited on May 15, 2024).
443. *Moerland, T. M.* Model-based Reinforcement Learning: A Survey / T. M. Moerland, J. Broekens, C. M. Jonker // Foundations and Trends in Machine Learning. — 2023. — Vol. 16, no. 1. — P. 1—118.
444. MONet: Unsupervised Scene Decomposition and Representation [Электронный ресурс] / C. P. Burgess [et al.]. — 2019. — arXiv: 1901.11390 [cs.CV]. — URL: <https://arxiv.org/abs/1901.11390> (visited on May 15, 2024).
445. Monotonic value function factorisation for deep multi-agent reinforcement learning / T. Rashid [et al.] // The Journal of Machine Learning Research. — 2020. — Vol. 21, no. 1. — P. 7234—7284.
446. Monte-Carlo robot path planning / T. Dam [et al.] // IEEE Robotics and Automation Letters. — 2022. — Vol. 7, no. 4. — P. 11213—11220.
447. *Moore, A. W.* Prioritized Sweeping: Reinforcement Learning with Less Data and Less Time / A. W. Moore, C. G. Atkeson // Machine Learning. — 1993. — Vol. 13. — P. 103—130.
448. *Mracek, C. P.* Control designs for the nonlinear benchmark problem via the state-dependent Riccati equation method / C. P. Mracek, J. R. Cloutier // International Journal of Robust and Nonlinear Control. — 1998. — Vol. 8. — P. 401—433.
449. Muesli: Combining Improvements in Policy Optimization / M. Hessel [et al.] // Proceedings of the 38th International Conference on Machine Learning, PMLR. Vol. 139. — 2021. — P. 4214—4226.
450. Multi-Agent Path Finding with Prioritized Communication Learning / W. Li [et al.] // 2022 International Conference on Robotics and Automation (ICRA). — 2022. — P. 10695—10701.
451. Multi-agent pathfinding: Definitions, variants, and benchmarks / R. Stern [et al.] // Proceedings of the International Symposium on Combinatorial Search. Vol. 10. — 2019. — P. 151—158.
452. Multi-Level Discovery of Deep Options [Электронный ресурс] / R. Fox [et al.]. — 2017. — arXiv: 1703.08294 [cs.LG]. — URL: <https://arxiv.org/abs/1703.08294> (visited on May 15, 2024).
453. Multi-object representation learning with iterative variational inference / K. Greff [et al.] // International Conference on Machine Learning. — PMLR. 2019. — P. 2424—2433.
454. Multiset-Equivariant Set Prediction with Approximate Implicit Differentiation [Электронный ресурс] / Y. Zhang [et al.] // International Conference on Learning Representations. — 2022. — URL: <https://openreview.net/forum?id=5K7RRqZEjoS> (visited on May 15, 2024).

455. *Mur-Artal, R.* ORB-SLAM: A Versatile and Accurate Monocular SLAM System / R. Mur-Artal, J. M. M. Montiel, J. D. Tardós // IEEE Transactions on Robotics. — 2015. — Vol. 31. — P. 1147—1163.
456. *Muravyev, K.* Enhancing exploration algorithms for navigation with visual SLAM / K. Muravyev, A. Bokovoy, K. Yakovlev // Russian Conference on Artificial Intelligence. — Springer. 2021. — P. 197—212.
457. *Murray, M.* Following natural language instructions for household tasks with landmark guided search and reinforced pose adjustment / M. Murray, M. Cakmak // IEEE Robotics and Automation Letters. — 2022. — Vol. 7, no. 3. — P. 6870—6877.
458. *Narayan-Chen, A.* Collaborative dialogue in Minecraft / A. Narayan-Chen, P. Jayannavar, J. Hockenmaier // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. — 2019. — P. 5405—5415.
459. *Nawaz, F.* Multiagent, Multitarget Path Planning in Markov Decision Processes / F. Nawaz, M. Ornik // IEEE Transactions on Automatic Control. — 2023. — Vol. 68, no. 12. — P. 7560—7574.
460. Neural controlled differential equations for irregular time series / P. Kidger [et al.] // Advances in Neural Information Processing Systems. Vol. 33. — 2020. — P. 6696—6707.
461. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning / A. Nagabandi [et al.] // 2018 IEEE International Conference on Robotics and Automation (ICRA). — 2017. — P. 7559—7566.
462. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning / A. Nagabandi [et al.] // 2018 IEEE international conference on robotics and automation (ICRA). — IEEE. 2018. — P. 7559—7566.
463. Neural Ordinary Differential Equations [Электронный ресурс] / R. T. Q. Chen [et al.] // Advances in Neural Information Processing Systems. Vol. 31 / ed. by S. Bengio [et al.]. — Curran Associates, Inc., 2018. — URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf (visited on May 15, 2024).
464. Neural Production Systems: Learning Rule-Governed Visual Dynamics [Электронный ресурс] / A. Goyal [et al.]. — 2022. — arXiv: 2103.01937 [cs.AI]. — URL: <https://arxiv.org/abs/2103.01937> (visited on May 15, 2024).
465. Neural-symbolic learning and reasoning: A survey and interpretation / A. d. Garcez [et al.] // Neuro-Symbolic Artificial Intelligence: The State of the Art. Vol. 342. — IOS Press, 2021. — P. 1—51.
466. *Nouri, A.* Dimension reduction and its application to model-based exploration in continuous spaces / A. Nouri, M. L. Littman // Machine Learning. — 2010. — Vol. 81. — P. 85—98.
467. Object discovery and representation networks / O. J. Hénaff [et al.] // European Conference on Computer Vision. — Springer. 2022. — P. 123—143.
468. Object goal navigation using goal-oriented semantic exploration / D. S. Chaplot [et al.] // Advances in Neural Information Processing Systems. Vol. 33. — 2020. — P. 4247—4258.

469. Object-Centric Diagnosis of Visual Reasoning [Электронный ресурс] / J. Yang [et al.]. — 2020. — arXiv: 2012.11587 [cs.CV]. — URL: <https://arxiv.org/abs/2012.11587> (visited on May 15, 2024).
470. Object-Centric Learning of Neural Policies for Zero-shot Transfer over Domains with Varying Quantities of Interest [Электронный ресурс] / V. Sharma [et al.] // PRL Workshop Series {\textendash} Bridging the Gap Between AI Planning and Reinforcement Learning. — 2023. — URL: <https://openreview.net/forum?id=mQtyk75pYZ> (visited on May 15, 2024).
471. Object-centric learning with slot attention / F. Locatello [et al.] // Advances in neural information processing systems. Vol. 33. — 2020. — P. 11525–11538.
472. ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects [Электронный ресурс] / D. Batra [et al.]. — 2020. — arXiv: 2006.13171 [cs.CV]. — URL: <https://arxiv.org/abs/2006.13171> (visited on May 15, 2024).
473. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems [Электронный ресурс] / S. Levine [et al.]. — 2020. — arXiv: 2005.01643 [cs.LG]. — URL: <https://arxiv.org/abs/2005.01643> (visited on May 15, 2024).
474. Offline Visual Representation Learning for Embodied Navigation [Электронный ресурс] / K. Yadav [et al.] // Workshop on Reincarnating Reinforcement Learning at ICLR 2023. — 2023. — URL: https://openreview.net/forum?id=Spfbts_vNY (visited on May 15, 2024).
475. *Oh, J.* Value Prediction Network [Электронный ресурс] / J. Oh, S. Singh, H. Lee // Advances in Neural Information Processing Systems. Vol. 30 / ed. by I. Guyon [et al.]. — Curran Associates, Inc., 2017. — URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/ffbd6cbb019a1413183c8d08f2929307-Paper.pdf (visited on May 15, 2024).
476. *Oh, J.* Value Prediction Network [Электронный ресурс] / J. Oh, S. Singh, H. Lee // Advances in Neural Information Processing Systems. Vol. 30 / ed. by I. Guyon [et al.]. — Curran Associates, Inc., 2017. — URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/ffbd6cbb019a1413183c8d08f2929307-Paper.pdf (visited on May 15, 2024).
477. On grounded planning for embodied tasks with language models / B. Y. Lin [et al.] // Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 37. — 2023. — P. 13192–13200.
478. On the Robustness of ChatGPT: An Adversarial and Out-of-distribution Perspective [Электронный ресурс] / J. Wang [et al.] // ICLR 2023 Workshop on Trustworthy and Reliable Large-Scale Machine Learning Models. — 2023. — URL: <https://openreview.net/forum?id=uw6HSkgom29> (visited on May 15, 2024).
479. *Oord, A. van den.* Neural Discrete Representation Learning [Электронный ресурс] / A. van den Oord, O. Vinyals, k. kavukcuoglu koray // Advances in Neural Information Processing Systems. Vol. 30 / ed. by I. Guyon [et al.]. — Curran Associates, Inc., 2017. — URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf (visited on May 15, 2024).
480. Open-vocabulary Object Detection via Vision and Language Knowledge Distillation [Электронный ресурс] / X. Gu [et al.] // International Conference on Learning Representations. — 2022. — URL: <https://openreview.net/forum?id=1L3lnMbR4WU> (visited on May 15, 2024).

481. OpenAI Gym [Электронный ресурс] / G. Brockman [et al.]. — 2016. — arXiv: 1606.01540 [cs.LG]. — URL: <https://arxiv.org/abs/1606.01540> (visited on May 15, 2024).
482. OPT: Open Pre-trained Transformer Language Models [Электронный ресурс] / S. Zhang [et al.]. — 2022. — arXiv: 2205.01068 [cs.CL]. — URL: <https://arxiv.org/abs/2205.01068> (visited on May 15, 2024).
483. Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam / C. Campos [et al.] // IEEE Transactions on Robotics. — 2021. — Vol. 37, no. 6. — P. 1874—1890.
484. Osipov, G. S. Sign-based representation and word model of actor / G. S. Osipov // 2016 IEEE 8th International Conference on Intelligent Systems (IS) / ed. by R. Yager [et al.]. — IEEE, 2016. — P. 22—26.
485. Osipov, G. S. Signs-Based vs. Symbolic Models / G. S. Osipov // Advances in Artificial Intelligence and Soft Computing / ed. by G. Sidorov, S. N. Galicia-Haro. — Springer International Publishing, 2015. — P. 3—11.
486. Oudeyer, P.-Y. What is intrinsic motivation? A typology of computational approaches / P.-Y. Oudeyer, F. Kaplan // Frontiers in neurorobotics. — 2007. — Vol. 1. — P. 6.
487. Palm: Scaling language modeling with pathways / A. Chowdhery [et al.] // Journal of Machine Learning Research. — 2023. — Vol. 24, no. 240. — P. 1—113.
488. Paraense, A. L. O. A machine consciousness approach to urban traffic control / A. L. O. Paraense, K. Raizer, R. R. Gudwin // Biologically Inspired Cognitive Architectures. — 2016. — Vol. 15. — P. 61—73.
489. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age / C. Cadena [et al.] // IEEE Transactions on robotics. — 2016. — Vol. 32, no. 6. — P. 1309—1332.
490. Path planning based on clothoid for autonomous valet parking / S. Liyang [et al.] // 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP). — IEEE. 2020. — P. 389—393.
491. Pathak, D. Self-supervised exploration via disagreement / D. Pathak, D. Gandhi, A. Gupta // International conference on machine learning. — PMLR. 2019. — P. 5062—5071.
492. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization / J. Zhang [et al.] // International Conference on Machine Learning. — PMLR. 2020. — P. 11328—11339.
493. Personal autonomy rehabilitation in home environments by a portable assistive robot / A. J. Huete [et al.] // IEEE Transactions on systems, man, and cybernetics, part c (applications and reviews). — 2011. — Vol. 42, no. 4. — P. 561—570.
494. PG-RRT: A gaussian mixture model driven, kinematically constrained bi-directional RRT for robot path planning / P. Sharma [et al.] // 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE. 2021. — P. 3666—3673.
495. Piaget, J. Le développement des mécanismes de la perception / J. Piaget // Bulletin de psychologie. — 1961. — Vol. 14, no. 187. — P. 368—374.

496. *Pichotta, K.* Learning statistical scripts with LSTM recurrent neural networks / K. Pichotta, R. Mooney // Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 30. — 2016. — P. 2800—2806.
497. Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control [Электронный ресурс] / K. Lowrey [et al.] // International Conference on Learning Representations. — 2019. — URL: <https://openreview.net/forum?id=Byey7n05FQ> (visited on May 15, 2024).
498. Planning to explore via self-supervised world models / R. Sekar [et al.] // International conference on machine learning. — PMLR. 2020. — P. 8583—8592.
499. Planning to explore via self-supervised world models / R. Sekar [et al.] // International Conference on Machine Learning. — PMLR. 2020. — P. 8583—8592.
500. Playing Atari with Deep Reinforcement Learning [Электронный ресурс] / V. Mnih [et al.]. — 2013. — arXiv: 1312.5602 [cs.LG]. — URL: <https://arxiv.org/abs/1312.5602> (visited on May 15, 2024).
501. POGEMA: partially observable grid environment for multiple agents [Электронный ресурс] / A. Skrynnik [et al.]. — 2022. — URL: <http://arxiv.org/abs/2206.10944> (visited on May 15, 2024).
502. Policy gradient methods for reinforcement learning with function approximation [Электронный ресурс] / R. S. Sutton [et al.] // Advances in neural information processing systems. Vol. 12. — 1999. — URL: https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf (visited on May 15, 2024).
503. Policy Gradient Search: Online Planning and Expert Iteration without Search Trees [Электронный ресурс] / T. Anthony [et al.]. — 2019. — arXiv: 1904.03646 [cs.LG]. — URL: <https://arxiv.org/abs/1904.03646> (visited on May 15, 2024).
504. Positioning and pressing elevator button by binocular vision and robot manipulator / F. Tiyu [et al.] // 2018 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC). — IEEE. 2018. — P. 120—133.
505. *Pospelov, D. A.* Knowledge in semiotic models / D. A. Pospelov, G. S. Osipov // Proceedings of the Second Workshop on Applied Semiotics, Seventh International Conference on Artificial Intelligence and Information-Control Systems of Robots (AIICSR'97). — Bratislava, 1997. — P. 1—12.
506. Predicting the Present and Future States of Multi-agent Systems from Partially-observed Visual Data [Электронный ресурс] / C. Sun [et al.] // International Conference on Learning Representations. — 2019. — URL: <https://openreview.net/forum?id=r1xdH3CcKX> (visited on May 15, 2024).
507. PRIMAL _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong / M. Damani [et al.] // IEEE Robotics and Automation Letters. — 2021. — Vol. 6, no. 2. — P. 2666—2673.
508. Primal: Pathfinding via reinforcement and imitation multi-agent learning / G. Sartoretti [et al.] // IEEE Robotics and Automation Letters. — 2019. — Vol. 4, no. 3. — P. 2378—2385.

509. Probabilistic inference for determining options in reinforcement learning / C. Daniel [et al.] // Machine Learning. — 2016. — Vol. 104. — P. 337—357.
510. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation / M. Deitke [et al.] // Advances in Neural Information Processing Systems. Vol. 35. — 2022. — P. 5982—5994.
511. Progprompt: Generating situated robot task plans using large language models / I. Singh [et al.] // 2023 IEEE International Conference on Robotics and Automation (ICRA). — IEEE. 2023. — P. 11523—11530.
512. Proximal Policy Optimization Algorithms [Электронный ресурс] / J. Schulman [et al.]. — 2017. — arXiv: 1707.06347. — URL: <http://arxiv.org/abs/1707.06347> (visited on May 15, 2024).
513. Puterman, M. L. Markov Decision Processes: Discrete Stochastic Dynamic Programming / M. L. Puterman. — Wiley-Interscience, 2005. — 392 p.
514. Puterman, M. L. Modified Policy Iteration Algorithms for Discounted Markov Decision Problems / M. L. Puterman, M. C. Shin // Management Science. — 1978. — Vol. 24. — P. 1127—1137.
515. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement Learning / T. Rashid [et al.] // 35th International Conference on Machine Learning, ICML 2018. Vol. 10. — 2018. — P. 6846—6859.
516. QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation [Электронный ресурс] / D. Kalashnikov [et al.]. — 2018. — arXiv: 1806.10293 [cs.LG]. — URL: <https://arxiv.org/abs/1806.10293> (visited on May 15, 2024).
517. R3M: A Universal Visual Representation for Robot Manipulation / S. Nair [et al.] // Proceedings of The 6th Conference on Robot Learning. Vol. 205 / ed. by K. Liu, D. Kulic, J. Ichnowski. — PMLR, 2023. — P. 892—909. — (Proceedings of Machine Learning Research).
518. Rainbow: Combining Improvements in Deep Reinforcement Learning / M. Hessel [et al.] // The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18). — 2018. — P. 3215—3222.
519. Rankooh, M. F. ITSAT: An Efficient SAT-Based Temporal Planner / M. F. Rankooh, G. Ghassem-Sani // Journal of Artificial Intelligence Research. — 2015. — Vol. 53. — P. 541—632.
520. Real-time loop closure in 2D LIDAR SLAM / W. Hess [et al.] // 2016 IEEE international conference on robotics and automation (ICRA). — IEEE. 2016. — P. 1271—1278.
521. Real-Time Motion Planning With Applications to Autonomous Urban Driving / Y. Kuwata [et al.] // IEEE Transactions on Control Systems Technology. — 2009. — Vol. 17. — P. 1105—1118.
522. Recurrent Environment Simulators [Электронный ресурс] / S. Chiappa [et al.] // International Conference on Learning Representations. — 2017. — URL: <https://openreview.net/forum?id=B1s6xvqlx> (visited on May 15, 2024).
523. Recurrent Experience Replay in Distributed Reinforcement Learning [Электронный ресурс] / S. Kapturowski [et al.] // International Conference on Learning Representations. — 2019. — URL: <https://openreview.net/forum?id=r1lyTjAqYX> (visited on May 15, 2024).

524. *Reddy, S.* SQIL: Imitation Learning via Reinforcement Learning with Sparse Rewards [Электронный ресурс] / S. Reddy, A. D. Dragan, S. Levine // International Conference on Learning Representations. — 2020. — URL: <https://openreview.net/forum?id=S1xKd24twB> (visited on May 15, 2024).
525. *Redmon, J.* YOLO9000: better, faster, stronger / J. Redmon, A. Farhadi // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2017. — P. 7263—7271.
526. *Reimers, N.* Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks / N. Reimers, I. Gurevych // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). — 2019. — P. 3982—3992.
527. Reinforcement Learning with Unsupervised Auxiliary Tasks [Электронный ресурс] / M. Jaderberg [et al.] // International Conference on Learning Representations. — 2017. — URL: <https://openreview.net/forum?id=SJ6yPD5xg> (visited on May 15, 2024).
528. Relational inductive biases, deep learning, and graph networks [Электронный ресурс] / P. W. Battaglia [et al.]. — 2018. — arXiv: 1806.01261 [cs.LG]. — URL: <https://arxiv.org/abs/1806.01261> (visited on May 15, 2024).
529. Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions [Электронный ресурс] / S. van Steenkiste [et al.] // International Conference on Learning Representations. — 2018. — URL: <https://openreview.net/forum?id=ryH20GbRW> (visited on May 15, 2024).
530. Revisiting Discrete Soft Actor-Critic [Электронный ресурс] / H. Zhou [et al.]. — 2023. — arXiv: 2209.10081 [cs.LG]. — URL: <https://arxiv.org/abs/2209.10081> (visited on May 15, 2024).
531. *Rezende, D. J.* Stochastic backpropagation and variational inference in deep latent gaussian models / D. J. Rezende, S. Mohamed, D. Wierstra // International conference on machine learning. Vol. 32. — 2014. — P. 1278—1286.
532. *Richard S. Sutton and Andrew G. Barto.* Reinforcement Learning. An Introduction / Richard S. Sutton and Andrew G. Barto. — MIT Press, 2018. — 550 p.
533. *Richter, S.* The LAMA planner: Guiding cost-based anytime planning with landmarks / S. Richter, M. Westphal // Journal of Artificial Intelligence Research. — 2010. — Vol. 39. — P. 127—177.
534. *Riemer, M.* Learning Abstract Options [Электронный ресурс] / M. Riemer, M. Liu, G. Tesauro // Advances in Neural Information Processing Systems. Vol. 31 / ed. by S. Bengio [et al.]. — Curran Associates, Inc., 2018. — URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/cdf28f8b7d14ab02d12a2329d71e4079-Paper.pdf (visited on May 15, 2024).
535. *Rivera, N.* Grid pathfinding on the 2k neighborhoods / N. Rivera, C. Hernández, J. Baier // Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 31. — 2017. — P. 891—897.
536. RL-RRT: Kinodynamic motion planning via learning reachability estimators from RL policies / H.-T. L. Chiang [et al.] // IEEE Robotics and Automation Letters. — 2019. — Vol. 4, no. 4. — P. 4298—4305.

537. RoBERTa: A Robustly Optimized BERT Pretraining Approach [Электронный ресурс] / Y. Liu [et al.]. — 2019. — arXiv: 1907.11692 [cs.CL]. — URL: <https://arxiv.org/abs/1907.11692> (visited on May 15, 2024).
538. Robot navigation in constrained pedestrian environments using reinforcement learning / C. Pérez-D'Arpino [et al.] // 2021 IEEE International Conference on Robotics and Automation (ICRA). — IEEE. 2021. — P. 1140—1146.
539. Rozenberszki, D. LOL: Lidar-only odometry and localization in 3D point cloud maps / D. Rozenberszki, A. L. Majdik // 2020 IEEE International Conference on Robotics and Automation (ICRA). — IEEE. 2020. — P. 4379—4385.
540. Rubanova, Y. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series [Электронный ресурс] / Y. Rubanova, R. T. Q. Chen, D. K. Duvenaud // Advances in Neural Information Processing Systems. Vol. 32 / ed. by H. Wallach [et al.]. — Curran Associates, Inc., 2019. — URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/42a6845a557bef704ad8ac9cb4461d43-Paper.pdf (visited on May 15, 2024).
541. Ryan, R. M. Intrinsic and extrinsic motivations: Classic definitions and new directions / R. M. Ryan, E. L. Deci // Contemporary educational psychology. — 2000. — Vol. 25, no. 1. — P. 54—67.
542. S3vae: Self-supervised sequential vae for representation disentanglement and data generation / Y. Zhu [et al.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2020. — P. 6538—6547.
543. Sample efficient reinforcement learning via model-ensemble exploration and exploitation / Y. Yao [et al.] // 2021 IEEE International Conference on Robotics and Automation (ICRA). — IEEE. 2021. — P. 4202—4208.
544. Sample factory: Egocentric 3d control from pixels at 100000 fps with asynchronous reinforcement learning / A. Petrenko [et al.] // International Conference on Machine Learning. — PMLR. 2020. — P. 7652—7662.
545. Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion [Электронный ресурс] / J. Buckman [et al.] // Advances in Neural Information Processing Systems. Vol. 31 / ed. by S. Bengio [et al.]. — Curran Associates, Inc., 2018. — URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/f02208a057804ee16ac72ff4d3cec53b-Paper.pdf (visited on May 15, 2024).
546. Samsonovich, A. V. Comparative analysis of implemented cognitive architectures / A. V. Samsonovich. — 2011.
547. Samsonovich, A. V. Emotional biologically inspired cognitive architecture / A. V. Samsonovich // Biologically Inspired Cognitive Architectures. — 2013. — Vol. 6. — P. 109—125.
548. Sancaktar, C. Curious exploration via structured world models yields zero-shot object manipulation / C. Sancaktar, S. Blaes, G. Martius // Advances in Neural Information Processing Systems. Vol. 35. — 2022. — P. 24170—24183.
549. Santosh, D. Autonomous image-based exploration for mobile robot navigation / D. Santosh, S. Achar, C. Jawahar // 2008 IEEE international conference on robotics and automation. — IEEE. 2008. — P. 2717—2722.

550. Savi++: Towards end-to-end object-centric learning from real-world videos / G. Elsayed [et al.] // Advances in Neural Information Processing Systems. Vol. 35. — 2022. — P. 28940—28954.
551. Scalable deep reinforcement learning for vision-based robotic manipulation / D. Kalashnikov [et al.] // Conference on Robot Learning. — PMLR. 2018. — P. 651—673.
552. Scaling instruction-finetuned language models / H. W. Chung [et al.] // Journal of Machine Learning Research. — 2024. — Vol. 25, no. 70. — P. 1—53.
553. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics / K. Kansky [et al.] // International conference on machine learning. — PMLR. 2017. — P. 1809—1818.
554. Schmidhuber, J. Deep Learning in Neural Networks: An Overview / J. Schmidhuber // Neural Networks. — 2015. — Vol. 61. — P. 85—117.
555. Schwarting, W. Planning and Decision-Making for Autonomous Vehicles / W. Schwarting, J. Alonso-Mora, D. Rus // Annual Review of Control, Robotics, and Autonomous Systems. — 2018. — Vol. 1. — P. 187—210.
556. SCRIMP: Scalable Communication for Reinforcement- and Imitation-Learning-Based Multi-Agent Pathfinding / Y. Wang [et al.] // Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems. — 2023. — P. 2598—2600. — (Visited on July 6, 2023).
557. Sebastian Thrun, W. B. Probabilistic Robotics / W. B. Sebastian Thrun, D. Fox. — 2005. — 647 p.
558. Sedighi, S. Guided hybrid A-star path planning algorithm for valet parking applications / S. Sedighi, D.-V. Nguyen, K.-D. Kuhnert // 2019 5th international conference on control, automation and robotics (ICCAR). — IEEE. 2019. — P. 570—575.
559. Sequencing the Connectome / A. M. Zador [et al.] // PLoS Biology. — 2012. — Vol. 10. — e1001411.
560. Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects [Электронный ресурс] / A. Kosioruk [et al.] // Advances in Neural Information Processing Systems. Vol. 31 / ed. by S. Bengio [et al.]. — Curran Associates, Inc., 2018. — URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/7417744a2bac776fabe5a09b21c707a2-Paper.pdf (visited on May 15, 2024).
561. Shan, T. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain / T. Shan, B. Englot // 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE. 2018. — P. 4758—4765.
562. Shapestacks: Learning vision-based physical intuition for generalised object stacking / O. Groth [et al.] // Proceedings of the european conference on computer vision (eccv). — 2018. — P. 702—717.
563. Shi, Z. Learning to Execute Actions or Ask Clarification Questions / Z. Shi, Y. Feng, A. Lipani // Findings of the Association for Computational Linguistics: NAACL 2022. — 2022. — P. 2060—2070.
564. Shoham, Y. Solving Sokoban with forward-backward reinforcement learning / Y. Shoham, G. Elidan // Proceedings of the International Symposium on Combinatorial Search. Vol. 12. — 2021. — P. 191—193.

565. *Shridhar, M.* Cliport: What and where pathways for robotic manipulation / M. Shridhar, L. Manuelli, D. Fox // Conference on Robot Learning. — PMLR. 2022. — P. 894—906.
566. *Shyam, P.* Model-based active exploration / P. Shyam, W. Jaśkowski, F. Gomez // International conference on machine learning. — PMLR. 2019. — P. 5779—5788.
567. *Silver, D.* Monte-Carlo tree search and rapid action value estimation in computer Go / D. Silver, S. Gelly // Artificial Intelligence. — 2011. — Vol. 175, no. 11. — P. 1856—1875.
568. *Silver, D.* Sample-based learning and search with permanent and transient memories / D. Silver, R. S. Sutton, M. Müller // Proceedings of the 25th international conference on Machine learning. — 2008. — P. 968—975.
569. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? [Электронный ресурс] / A. Kadian [et al.]. — 2020. — arXiv: 1912 . 06321 [cs.CV]. — URL: <https://arxiv.org/abs/1912.06321> (visited on May 15, 2024).
570. *Simonovsky, M.* Graphvae: Towards generation of small graphs using variational autoencoders / M. Simonovsky, N. Komodakis // Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4–7, 2018, Proceedings, Part I 27. — Springer. 2018. — P. 412—422.
571. Simple but effective: Clip embeddings for embodied ai / A. Khandelwal [et al.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2022. — P. 14829—14838.
572. *Şimşek, Ö.* Identifying useful subgoals in reinforcement learning by local graph partitioning / Ö. Şimşek, A. P. Wolfe, A. G. Barto // Proceedings of the 22nd international conference on Machine learning. — 2005. — P. 816—823.
573. *Singh, G.* Illiterate DALL-E Learns to Compose [Электронный ресурс] / G. Singh, F. Deng, S. Ahn // International Conference on Learning Representations. — 2022. — URL: <https://openreview.net/forum?id=h00YV0We3oh> (visited on May 15, 2024).
574. *Singh, S.* Reinforcement Learning with Soft State Aggregation [Электронный ресурс] / S. Singh, T. Jaakkola, M. Jordan // Advances in Neural Information Processing Systems. Vol. 7 / ed. by G. Tesauro, D. Touretzky, T. Leen. — MIT Press, 1994. — URL: https://proceedings.neurips.cc/paper_files/paper/1994/file/287e03db1d99e0ec2edb90d079e142f3-Paper.pdf (visited on May 15, 2024).
575. SlotFormer: Unsupervised Visual Dynamics Simulation with Object-Centric Models [Электронный ресурс] / Z. Wu [et al.] // The Eleventh International Conference on Learning Representations. — 2023. — URL: <https://openreview.net/forum?id=TFbwV6I0VLg> (visited on May 15, 2024).
576. *Smith, L. N.* Super-convergence: Very fast training of neural networks using large learning rates / L. N. Smith, N. Topin // Artificial intelligence and machine learning for multi-domain operations applications. Vol. 11006. — 2019. — P. 369—386.
577. *Snaider, J.* Vector LIDA / J. Snaider, S. Franklin // Procedia Computer Science. — 2014. — Vol. 41. — P. 188—203.

578. Socratic Models: Composing Zero-Shot Multimodal Reasoning with Language [Электронный ресурс] / A. Zeng [et al.] // The Eleventh International Conference on Learning Representations. — 2023. — URL: <https://openreview.net/forum?id=G2Q2Mh3avow> (visited on May 15, 2024).
579. Soetanto, D. Adaptive, non-singular path-following control of dynamic wheeled robots / D. Soetanto, L. Lapierre, A. Pascoal // 42nd IEEE international conference on decision and control (IEEE Cat. No. 03CH37475). Vol. 2. — IEEE. 2003. — P. 1765—1770.
580. Soft Actor-Critic Algorithms and Applications [Электронный ресурс] / T. Haarnoja [et al.]. — 2019. — arXiv: 1812.05905 [cs.LG]. — URL: <https://arxiv.org/abs/1812.05905> (visited on May 15, 2024).
581. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor / T. Haarnoja [et al.] // International conference on machine learning. — PMLR. 2018. — P. 1861—1870.
582. SOLOv2: Dynamic and Fast Instance Segmentation / X. Wang [et al.] // Advances in Neural Information Processing Systems. Vol. 33. — 2020. — P. 17721—17732.
583. Solving the Rubik’s cube with deep reinforcement learning and search / F. Agostinelli [et al.] // Nature Machine Intelligence. — 2019. — Vol. 1, no. 8. — P. 356—363.
584. SPACE: Unsupervised Object-Oriented Scene Representation via Spatial Attention and Decomposition [Электронный ресурс] / Z. Lin [et al.] // International Conference on Learning Representations. — 2020. — URL: <https://openreview.net/forum?id=rk103ySYDH> (visited on May 15, 2024).
585. Spatial Broadcast Decoder: A Simple Architecture for Learning Disentangled Representations in VAEs [Электронный ресурс] / N. Watters [et al.]. — 2019. — arXiv: 1901.07017 [cs.LG]. — URL: <https://arxiv.org/abs/1901.07017> (visited on May 15, 2024).
586. Spelke, E. S. Core knowledge. / E. S. Spelke, K. D. Kinzler // Developmental science. — 2007. — Vol. 101. — P. 89—96.
587. Spice: Semantic propositional image caption evaluation / P. Anderson [et al.] // Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part V 14. — Springer. 2016. — P. 382—398.
588. Splitnet: Sim2sim and task2task transfer for embodied visual navigation / D. Gordon [et al.] // Proceedings of the IEEE/CVF International Conference on Computer Vision. — 2019. — P. 1022—1031.
589. Spotlight Attention: Robust Object-Centric Learning With a Spatial Locality Prior [Электронный ресурс] / A. Chakravarthy [et al.]. — 2023. — arXiv: 2305.19550 [cs.CV]. — URL: <https://arxiv.org/abs/2305.19550> (visited on May 15, 2024).
590. Ssd: Single shot multibox detector / W. Liu [et al.] // Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. — Springer. 2016. — P. 21—37.
591. Stable-baselines3: Reliable reinforcement learning implementations / A. Raffin [et al.] // The Journal of Machine Learning Research. — 2021. — Vol. 22, no. 1. — P. 12348—12355.

592. *Standley, T.* Finding optimal solutions to cooperative pathfinding problems / T. Standley // Proceedings of the AAAI conference on artificial intelligence. Vol. 24. — 2010. — P. 173—178.
593. *Stanić, A.* R-SQAIR: Relational Sequential Attend, Infer, Repeat [Электронный ресурс] / A. Stanić, J. Schmidhuber. — 2019. — arXiv: 1910.05231 [cs.LG]. — URL: <https://arxiv.org/abs/1910.05231> (visited on May 15, 2024).
594. State Representation Learning for Control: An Overview / T. Lesort [et al.] // Neural networks. — 2018. — Vol. 108. — P. 379—392.
595. *Stewart, T. C.* Spaun: A Perception-Cognition-Action Model Using Spiking Neurons / T. C. Stewart, F.-X. Choo, C. Eliasmith // Cognitive Science. — 2012. — Vol. 34. — P. 1018—1023.
596. *Stolle, M.* Learning options in reinforcement learning / M. Stolle, D. Precup // Abstraction, Reformulation, and Approximation: 5th International Symposium, SARA 2002 Kananaskis, Alberta, Canada August 2–4, 2002 Proceedings 5. — Springer. 2002. — P. 212—223.
597. *Sumikura, S.* OpenVSLAM: A versatile visual SLAM framework / S. Sumikura, M. Shibuya, K. Sakurada // Proceedings of the 27th ACM International Conference on Multimedia. — 2019. — P. 2292—2295.
598. *Sun, R.* Modeling meta-cognition in a cognitive architecture / R. Sun, X. Zhang, R. Mathews // Cognitive Systems Research. — 2006. — Vol. 7, no. 4. — P. 327—338.
599. *Sutton, R. S.* Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning / R. S. Sutton, D. Precup, S. Singh // Artificial intelligence. — 1999. — Vol. 112, no. 1/2. — P. 181—211.
600. *Sutton, R. S.* Dyna, an integrated architecture for learning, planning, and reacting / R. S. Sutton // SIGART Bull. — 1991. — Vol. 2. — P. 160—163.
601. *Sutton, R. S.* Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning / R. S. Sutton, D. Precup, S. Singh // Artificial Intelligence. — 1999. — Vol. 112. — P. 181—211.
602. Tagger: Deep Unsupervised Perceptual Grouping [Электронный ресурс] / K. Greff [et al.] // Advances in Neural Information Processing Systems. Vol. 29 / ed. by D. Lee [et al.]. — Curran Associates, Inc., 2016. — URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/01eee509ee2f68dc6014898c309e86bf-Paper.pdf (visited on May 15, 2024).
603. *Talvitie, E.* Self-correcting models for model-based reinforcement learning / E. Talvitie // Proceedings of the AAAI conference on artificial intelligence. Vol. 31. — 2017. — P. 2597—2603.
604. TAPEX: Table Pre-training via Learning a Neural SQL Executor [Электронный ресурс] / Q. Liu [et al.] // International Conference on Learning Representations. — 2022. — URL: <https://openreview.net/forum?id=050443AsCP> (visited on May 15, 2024).
605. Target Entropy Annealing for Discrete Soft Actor-Critic [Электронный ресурс] / Y. Xu [et al.] // Deep RL Workshop NeurIPS 2021. — 2021. — URL: <https://openreview.net/forum?id=jJKzGBBQiZu> (visited on May 15, 2024).
606. Teach: Task-driven embodied agents that chat / A. Padmakumar [et al.] // Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 36. — 2022. — P. 2017—2025.

607. Tell, Draw, and Repeat: Generating and Modifying Images Based on Continual Linguistic Instruction / A. El-Nouby [et al.] // 2019 IEEE/CVF International Conference on Computer Vision (ICCV). — 2019. — P. 10303—10311.
608. *Tesauro, G.* On-line Policy Improvement using Monte-Carlo Search [Электронный ресурс] / G. Tesauro, G. Galperin // Advances in Neural Information Processing Systems. Vol. 9 / ed. by M. Mozer, M. Jordan, T. Petsche. — MIT Press, 1996. — URL: https://proceedings.neurips.cc/paper_files/paper/1996/file/996009f2374006606f4c0b0fda878af1-Paper.pdf (visited on May 15, 2024).
609. The arcade learning environment: An evaluation platform for general agents / M. G. Bellemare [et al.] // Journal of Artificial Intelligence Research. — 2013. — Vol. 47. — P. 253—279.
610. The Navigation and Control technology inside the AR.Drone micro UAV / P.-J. Bristeau [et al.] // IFAC Proceedings Volumes. — 2011. — Vol. 44. — P. 1477—1484.
611. The opencog framework / B. Goertzel [et al.] // Engineering General Intelligence, Part 2: The CogPrime Architecture for Integrative, Embodied AGI. — 2014. — P. 3—29.
612. The Pile: An 800GB Dataset of Diverse Text for Language Modeling [Электронный ресурс] / L. Gao [et al.]. — 2020. — arXiv: 2101.00027 [cs.CL]. — URL: <https://arxiv.org/abs/2101.00027> (visited on May 15, 2024).
613. The predictron: End-to-end learning and planning / D. Silver [et al.] // International Conference on Machine Learning. — PMLR. 2017. — P. 3191—3199.
614. The surprising effectiveness of ppo in cooperative multi-agent games / C. Yu [et al.] // Advances in Neural Information Processing Systems. Vol. 35. — 2022. — P. 24611—24624.
615. The unsurprising effectiveness of pre-trained vision models for control / S. Parisi [et al.] // international conference on machine learning. — PMLR. 2022. — P. 17359—17371.
616. The value equivalence principle for model-based reinforcement learning / C. Grimm [et al.] // Advances in Neural Information Processing Systems. Vol. 33. — Curran Associates, Inc., 2020. — P. 5541—5552.
617. Theta*: Any-angle path planning on grids / K. Daniel [et al.] // Journal of Artificial Intelligence Research. — 2010. — Vol. 39. — P. 533—579.
618. *Thrun, S.* Finding Structure in Reinforcement Learning [Электронный ресурс] / S. Thrun, A. Schwartz // Advances in Neural Information Processing Systems. Vol. 7 / ed. by G. Tesauro, D. Touretzky, T. Leen. — MIT Press, 1994. — URL: https://proceedings.neurips.cc/paper_files/paper/1994/file/7ce3284b743aefde80ffd9aec500e085-Paper.pdf (visited on May 15, 2024).
619. *Todorov, E.* A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems / E. Todorov, W. Li // Proceedings of the 2005, American Control Conference. Vol. 1. — 2005. — P. 300—306.
620. Towards Healthy AI: Large Language Models Need Therapists Too [Электронный ресурс] / B. Lin [et al.]. — 2023. — arXiv: 2304.00416 [cs.AI]. — URL: <https://arxiv.org/abs/2304.00416> (visited on May 15, 2024).

621. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback [Электронный ресурс] / Y. Bai [et al.]. — 2022. — arXiv: 2204.05862 [cs.CL]. — URL: <https://arxiv.org/abs/2204.05862> (visited on May 15, 2024).
622. Training language models to follow instructions with human feedback / L. Ouyang [et al.] // Advances in Neural Information Processing Systems. Vol. 35. — 2022. — P. 27730—27744.
623. Transporter networks: Rearranging the visual world for robotic manipulation / A. Zeng [et al.] // Conference on Robot Learning. — PMLR. 2021. — P. 726—747.
624. TreeQN and ATreeC: Differentiable Tree Planning for Deep Reinforcement Learning [Электронный ресурс] / G. Farquhar [et al.] // International Conference on Learning Representations. — 2018. — URL: <https://openreview.net/forum?id=H1dh6Ax0Z> (visited on May 15, 2024).
625. Trust Region Policy Optimization / J. Schulman [et al.] // Proceedings of The 32nd International Conference on Machine Learning. Vol. 37. — 2015. — P. 1889—1897.
626. *Tzafestas, S. G.* Introduction to mobile robot control / S. G. Tzafestas. — Elsevier, 2013. — 691 p.
627. *Ullman, T.* Large Language Models Fail on Trivial Alterations to Theory-of-Mind Tasks [Электронный ресурс] / T. Ullman. — 2023. — arXiv: 2302.08399 [cs.AI]. — URL: <https://arxiv.org/abs/2302.08399> (visited on May 15, 2024).
628. Unified Questioner Transformer for Descriptive Question Generation in Goal-Oriented Visual Dialogue / S. Matsumori [et al.] // 2021 IEEE/CVF International Conference on Computer Vision (ICCV). — 2021. — P. 1878—1887.
629. Universal Value Function Approximators / T. Schaul [et al.] // Proceedings of The 32nd International Conference on Machine Learning (ICML 2015). — 2015. — P. 1312—1320.
630. Unlocking slot attention by changing optimal transport costs / Y. Zhang [et al.] // International Conference on Machine Learning. — PMLR. 2023. — P. 41931—41951.
631. Unsupervised Learning of Object Keypoints for Perception and Control [Электронный ресурс] / T. D. Kulkarni [et al.] // Advances in Neural Information Processing Systems. Vol. 32 / ed. by H. Wallach [et al.]. — Curran Associates, Inc., 2019. — URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/dae3312c4c6c7000a37ecfb7b0aeb0e4-Paper.pdf (visited on May 15, 2024).
632. Unsupervised State Representation Learning in Atari [Электронный ресурс] / A. Anand [et al.] // Advances in Neural Information Processing Systems. Vol. 32 / ed. by H. Wallach [et al.]. — Curran Associates, Inc., 2019. — URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/6fb52e71b837628ac16539c1ff911667-Paper.pdf (visited on May 15, 2024).
633. Using deep reinforcement learning with automatic curriculum learning for mapless navigation in intralogistics / H. Xue [et al.] // Applied Sciences. — 2022. — Vol. 12, no. 6. — P. 3153.
634. Variational Option Discovery Algorithms [Электронный ресурс] / J. Achiam [et al.]. — 2018. — arXiv: 1807.10299 [cs.AI]. — URL: <https://arxiv.org/abs/1807.10299> (visited on May 15, 2024).
635. *Vedantam, R.* Cider: Consensus-based image description evaluation / R. Vedantam, C. Lawrence Zitnick, D. Parikh // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2015. — P. 4566—4575.

636. Video pretraining (vpt): Learning to act by watching unlabeled online videos / B. Baker [et al.] // Advances in Neural Information Processing Systems. Vol. 35. — 2022. — P. 24639—24654.
637. VIME: Variational Information Maximizing Exploration [Электронный ресурс] / R. Houthooft [et al.] // Advances in Neural Information Processing Systems. Vol. 29 / ed. by D. Lee [et al.]. — Curran Associates, Inc., 2016. — URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/abd815286ba1007abfbb8415b83ae2cf-Paper.pdf (visited on May 15, 2024).
638. Ving: Learning open-world navigation with visual goals / D. Shah [et al.] // 2021 IEEE International Conference on Robotics and Automation (ICRA). — IEEE. 2021. — P. 13215—13222.
639. Virtualhome: Simulating household activities via programs / X. Puig [et al.] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2018. — P. 8494—8502.
640. Vision-based SLAM using the Rao-Blackwellised particle filter / R. Sim [et al.] // IJCAI Workshop on Reasoning with Uncertainty in Robotics. Vol. 14. — Citeseer. 2005. — P. 9—16.
641. Visual room rearrangement / L. Weihs [et al.] // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. — 2021. — P. 5922—5931.
642. Volpi, N. C. Goal-directed empowerment: combining intrinsic motivation and task-oriented behavior / N. C. Volpi, D. Polani // IEEE Transactions on Cognitive and Developmental Systems. — 2020. — Vol. 15, no. 2. — P. 361—372.
643. Walk the Random Walk: Learning to Discover and Reach Goals Without Supervision [Электронный ресурс] / L. Mezghani [et al.] // ICLR Workshop on Agent Learning in Open-Endedness. — 2022. — URL: <https://openreview.net/forum?id=BeerQw-L-c> (visited on May 15, 2024).
644. Wang, F. Robot button pressing in human environments / F. Wang, G. Chen, K. Hauser // 2018 IEEE international conference on robotics and automation (ICRA). — IEEE. 2018. — P. 7173—7180.
645. Wang, Y. Meta-SAC: Auto-tune the Entropy Temperature of Soft Actor-Critic via Metagradient [Электронный ресурс] / Y. Wang, T. Ni. — 2020. — arXiv: 2007.01932 [cs.LG]. — URL: <https://arxiv.org/abs/2007.01932> (visited on May 15, 2024).
646. When to Trust Your Model: Model-Based Policy Optimization [Электронный ресурс] / M. Janner [et al.] // Advances in Neural Information Processing Systems. Vol. 32 / ed. by H. Wallach [et al.]. — Curran Associates, Inc., 2019. — URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/5faf461eff3099671ad63c6f3f094f7f-Paper.pdf (visited on May 15, 2024).
647. Wiering, M. Efficient model-based exploration / M. Wiering, J. Schmidhuber // Proceedings of the fifth international conference on simulation of adaptive behavior From animals to animats 5. — 1998. — P. 223—228.
648. Willemse, D. Value targets in off-policy AlphaZero: a new greedy backup / D. Willemse, H. Baier, M. Kaisers // Neural Computing and Applications. — 2021. — Vol. 34. — P. 1801—1814.
649. Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning / R. J. Williams // Machine learning. — 1992. — Vol. 8. — P. 229—256.

650. *Williams, R. J.* Simple statistical gradient-following algorithms for connectionist reinforcement learning / R. J. Williams // Machine Learning. — 1992. — Vol. 8. — P. 229—256.
651. *Yakovlev, K.* Grid-based angle-constrained path planning / K. Yakovlev, E. Baskin, I. Hramoin // KI 2015: Advances in Artificial Intelligence: 38th Annual German Conference on AI, Dresden, Germany, September 21-25, 2015, Proceedings 38. — Springer. 2015. — P. 208—221.
652. *Yildiz, C.* Continuous-time model-based reinforcement learning / C. Yildiz, M. Heinonen, H. Lähdesmäki // International Conference on Machine Learning. — PMLR. 2021. — P. 12009—12018.
653. Yolactedge: Real-time instance segmentation on the edge / H. Liu [et al.] // 2021 IEEE international conference on robotics and automation (ICRA). — IEEE. 2021. — P. 9579—9585.
654. *Zadaianchuk, A.* Self-supervised reinforcement learning with independently controllable subgoals / A. Zadaianchuk, G. Martius, F. Yang // Conference on Robot Learning. — PMLR. 2022. — P. 384—394.
655. *Zadaianchuk, A.* Self-supervised Visual Reinforcement Learning with Object-centric Representations [Электронный ресурс] / A. Zadaianchuk, M. Seitzer, G. Martius // International Conference on Learning Representations. — 2021. — URL: <https://openreview.net/forum?id=xppLmXCb0w1> (visited on May 15, 2024).
656. *Zerbel, N.* Multiagent monte carlo tree search / N. Zerbel, L. Yliniemi // Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. — 2019. — P. 2309—2311.
657. Zero-shot text-to-image generation / A. Ramesh [et al.] // International conference on machine learning. — Pmlr. 2021. — P. 8821—8831.
658. *Zhang, J.* Aerial and ground-based collaborative mapping: an experimental study / J. Zhang, S. Singh // Field and Service Robotics: Results of the 11th International Conference. — Springer. 2018. — P. 397—412.
659. *Zhang, Y.* Deep Set Prediction Networks [Электронный ресурс] / Y. Zhang, J. Hare, A. Prugel-Bennett // Advances in Neural Information Processing Systems. Vol. 32 / ed. by H. Wallach [et al.]. — Curran Associates, Inc., 2019. — URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/6e79ed05baec2754e25b4eac73a332d2-Paper.pdf (visited on May 15, 2024).
660. *Zhong, Y. D.* Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control [Электронный ресурс] / Y. D. Zhong, B. Dey, A. Chakraborty // International Conference on Learning Representations. — 2020. — URL: <https://openreview.net/forum?id=ryxmb1rKDS> (visited on May 15, 2024).
661. *Артемьев, Е. Ю.* Психология субъективной семантики / Е. Ю. Артемьева. — М. : Издательство ЛКИ, 2007. — 126 с.
662. *Выготский, Л. С.* Мысление и речь / Л. С. Выготский // Психология развития человека / под ред. С. Бобко. — Эксмо, 2005. — С. 664—1019.
663. *Дмитриев, М.* Гладкий нелинейный регулятор в слабо нелинейной системе управления с коэффициентами, зависящими от состояния / М. Дмитриев, Д. Макаров // Труды ИСА РАН. — 2014. — Вып. 4. — С. 3—13.

664. *Дмитриев, М. Г.* Итерационный алгоритм синтеза управления в сингулярно возмущенной нелинейной задаче на основе SDRE техники / М. Г. Дмитриев, Д. А. Макаров // Информационные технологии и вычислительные системы. — 2020. — № 1. — С. 76—84.
665. *Иваницкий, А. М.* Мозговая основа субъективных переживаний: гипотеза информационного синтеза / А. М. Иваницкий // Журнал высшей нервной деятельности. — 1996. — Т. 46, № 2. — С. 241—282.
666. Извлечение сценарной информации из текстов. Часть 1. Постановка задачи и обзор методов / М. И. Суворова [и др.] // Искусственный интеллект и принятие решений. — 2020. — № 1. — С. 17—26.
667. Интеллектуальное управление транспортными средствами: стандарты, проекты, реализации / Г. Осипов [и др.] // Авиакосмическое приборостроение. — 2009. — Т. 6. — С. 34—43.
668. *Карпов, В.* От колаборативной робототехники к социальным роботам для поддержки людей с ограниченными возможностями: новые направления разработки использования интеллектуальных агентов / В. Карпов, В. Тараков // Интеллектуальные технологии и средства реабилитации людей с ограниченными возможностями (ИТСР-2018). Труды III международной конференции. — Springer International Publishing, 2018. — С. 20—29.
669. *Кейслер, Г.* Теория моделей / Г. Кейслер, Ч. Чен. — М. : МИР, 1977. — 616 с.
670. *Киселев, Г. А.* Знаковый подход к задаче распределения ролей в коалиции когнитивных агентов / Г. А. Киселев, А. И. Панов // Труды СПИИРАН. — 2018. — Т. 2, № 57. — С. 161—187.
671. *Клини, С.* Математическая логика / С. Клини. — М. : МИР, 1973. — 480 с.
672. Коммуникативная грамматика русского языка / Г. А. Золотова [и др.]. — Издательский центр "Азбуковник", 2019. — 662 с.
673. *Леонтьев, А. Н.* Деятельность. Сознание. Личность / А. Н. Леонтьев. — М. : Политиздат, 1977. — 304 с.
674. Моделирование поведения, управляемого сознанием / Ю. М. Кузнецова [и др.] // Системный анализ и информационные технологии: тр. Четвертой Междунар. конф. (Абзаково, Россия, 17–23 авг. 2011 г.): в 2т. Т. 1. — Челябинск : Изд-во Челяб. Гос. ун-та, 2011. — С. 6—13.
675. *Осипов, Г. С.* Динамические интеллектуальные системы / Г. С. Осипов // Искусственный интеллект и принятие решений. — 2008. — № 1. — С. 47—54.
676. *Осипов, Г. С.* Методы искусственного интеллекта / Г. С. Осипов. — М. : ФИЗМАТЛИТ, 2015. — 297 с.
677. *Осипов, Г. С.* Управление поведением как функция сознания. I. Картина мира и целеполагание / Г. С. Осипов, А. И. Панов, Н. В. Чудова // Известия Российской академии наук. Теория и системы управления. — 2014. — № 4. — С. 49—62.
678. *Осипов, Г. С.* Прикладная семиотика / Г. С. Осипов, Д. А. Поспелов // Новости искусственного интеллекта. — 1999. — № 1. — С. 9—35.
679. *Поспелов, Д. А.* Десять «горячих точек» в исследованиях по искусственному интеллекту / Д. А. Поспелов // Искусственный интеллект и принятие решений. — 2019. — № 4. — С. 3—9.
680. *Рубинштейн, С. Я.* Экспериментальные методики патопсихологии и опыт применения их в клинике / С. Я. Рубинштейн. — Апрель-пресс, 2004. — 224 с.
681. *Рыбина, Г.* Методы и средства интеллектуального планирования: применение для управления процессами построения интегрированных экспертных систем / Г. Рыбина, Ю. Блохин // Искусственный интеллект и принятие решений. — 2015. — № 1. — С. 75—93.

682. Стабилизация нелинейных дискретных динамических систем с параметром и с коэффициентами, зависящими от состояния / С. В. Емельянов [и др.] // Доклады Академии наук. — 2016. — Т. 466, № 3. — С. 282—284.
683. *Стефанюк, В. Л.* Локальная организация интеллектуальных систем / В. Л. Стефанюк. — М. : ФИЗМАТЛИТ, 2004. — 328 с.
684. *Тарасов, В. Б.* От многоагентных систем к интеллектуальным организациям / В. Б. Тарасов. — М. : Эдиториал УРСС, 2002. — 352 с.
685. *Цетлин, М.* Исследования по теории автоматов и моделирование биологических систем / М. Цетлин. — М. : Наука, 1969. — 316 с.
686. *Чудова, Н. В.* Концептуальное описание картины мира для задачи моделирования поведения, основанного на сознании / Н. В. Чудова // Искусственный интеллект и принятие решений. — 2012. — № 2. — С. 51—62.
687. *Чудова, Н. В.* Психологические аспекты планирования в знаковой картине мира / Н. В. Чудова // Шестнадцатая Национальная конференция по искусственному интеллекту с международным участием КИИ-2018 (24–27 сентября 2018 г., г. Москва, Россия). Труды конференции. В 2-х томах. — М. : РКП, 2018. — С. 88—95.
688. *Чудова, Н. В.* Актуальные проблемы моделирования целеполагания в знаковой картине мира. Взгляд психолога / Н. В. Чудова // Искусственный интеллект и принятие решений. — 2020. — № 1. — С. 70—79.
689. *Чудова, Н. В.* Концептуальная модель самосознания для знаковой картины мира интеллектуального агента / Н. В. Чудова, Ю. М. Кузнецова // Искусственный интеллект и принятие решений. — 2018. — № 4. — С. 86—94.
690. *Эделмен, Д.* Разумный мозг / Д. Эделмен, В. Маунткасл ; под ред. Е. Н. Соколов ; пер. Н. Ю. Алексеенко. — М. : Мир, 1981. — 134 с.
691. *Яковлев, К.* Графовые модели в задаче планирования траектории на плоскости / К. Яковлев, Е. Баскин // Искусственный интеллект и принятие решений. — 2013. — Вып. 1. — С. 5—12.
692. *Яковлев, К.* Метод автоматического планирования траектории беспилотного летательного аппарата в условиях ограничений на динамику полета / К. Яковлев, Д. Макаров, Е. Баскин // Искусственный интеллект и принятие решений. — 2014. — Вып. 4. — С. 4—17.
693. *Яковлев, К. С.* Программный комплекс навигации и управления беспилотными транспортными средствами / К. С. Яковлев, А. В. Петров, В. В. Хитиков // Информационные технологии и вычислительные системы. — 2013. — № 3. — С. 72—83.

Список рисунков

1.1	Итеративный цикл взаимодействия агента (AI) и среды (Environment), включающий в себя получение информации из среды (State, Reward) и выполнение действия в ней (Action).	21
1.2	Виды аппроксимации функции полезности: слева отображение вида $S \rightarrow \mathbb{R}$, посередине — вида $S \times A \rightarrow \mathbb{R}$ и справа — вида $S \rightarrow \mathbb{R}^m$	29
1.3	Обработка высокоразмерного наблюдения в алгоритме глубокой Q-сети (DQN) [323] с помощью сверточной нейронной сети.	31
1.4	Визуальная интерпретация отношения правдоподобия при обновлении параметров стратегии в соответствии с полезностью траектории опыта.	34
1.5	Общая схема работы архитектуры «актор-критик»: обучение критика (Critic) по TD-ошибке функции полезности (Value Function) и обновление актора (Actor) с функцией стратегии (Policy Function).	37
1.6	Иллюстрация принципа, заложенного в теорему о нижней границе полезности стратегии. Функция $\eta(\theta)$ — истинная функция полезности, $L(\theta)$ — суррогатная функция.	40
1.7	Усеченная функция потерь L^{CLIP} для алгоритма PPO.	43
1.8	Пример переходов между состояниями при классическом планировании в задаче перемещения грузов погрузчиками с действиями выгрузить груз (inload), погрузить груз (load), переехать на новую площадку (move).	47
1.9	Интерпретация домена планирования как системы с переменными состояниями. .	51
1.10	Обобщенная схема работы методы обучения с подкреплением на основе модели с этапом планирования по модели.	59
1.11	Общая схема интеграции обучения и планирования (адаптировано из [443]). . . .	74
2.1	Взаимосвязь архитектур STRL и NSLP с программно-алгоритмическим инструментарием, представленным в дальнейших главах диссертационного исследования.	79
2.2	Основные функциональные блоки, характерные для многих когнитивных архитектур	81
2.3	Архитектура STRL системы управления группой сложных технических объектов.	87
2.4	Схема работы распознающего автомата (нижние индексы опущены). Из множества функций распознавания \hat{F}_i^j по управляющему вектору \hat{x}^{j+1} , поступающему с верхнего уровня иерархии, выбирается подмножество активных функций, по которому формируется множество активных матриц предсказания Z^* . Это множество фильтруется в каждый момент времени в соответствии с декартовым расстоянием до входного вектора \bar{x}^j . Также каждый момент времени вычисляются выходной вектор \bar{x}^* и управляющий вектор \hat{x}^j , отправляемый на нижний уровень иерархии.	91
2.5	Траектория, построенная алгоритмом LIAN.	95

2.6	К методу определения геометрических ограничений.	100
2.7	Основные компоненты и межмодульное взаимодействие в архитектуре STRL2	101
2.8	Взаимосвязь ранее рассматриваемых классов задач обучения принятию решений в условиях неопределенности в динамической среде с семействами методов, предназначенных для их решения. Архитектура NSLP предназначена для решения наиболее общего класса задач обучения выполнения языковых инструкций в среде с использованием нейросимвольной интеграции для повышения качества получаемых стратегий агента.	105
2.9	Архитектура NSLP агента с подсистемами обучения и планирования поведения. Выделены три составляющих архитектуры: концептуальный уровень планирования, сенсорный уровень одновременного обучения и планирования, а также уровень нейросимвольной интеграции с объектным представлением сцен. Красным цветом помечены сенсорные компоненты, синим — компоненты модели мира (объектной и латентной), зеленым — компоненты действия (план, стратегия, полезность действий), желтым — нейросимвольные интеграционные компоненты. Прямоугольниками обозначены внутренние представления, трапециями — обучаемые модели, прямоугольниками со скругленными углами — алгоритмические процедуры обучения или принятия решений.	108
2.10	Байесовские графические модели латентной динамики внешней среды. Первые два наблюдения (серые) подаются на модель, и она предсказывает третье наблюдение. Сплошные линии — процесс генерации, пунктирные — вывод модели. Слева представлена детерминированная модель с использованием рекуррентной нейронной сети, где h_t — скрытый вектор модели, моделирующий информационное состояние среды. Посередине представлен вариант со стохастическими состояниями s_t (SSM, space-state model, модель в пространстве состояний), где переходы полностью стохастичны. Справа — гибридный вариант с детерминированной и стохастической частью (RSSM, recurrent space-state model).	111
2.11	(а) — модельный пример перемещения агента по клеточной среде с комнатами, открывающимися проходами между ними и опасными состояниями, в которых завершается эпизод взаимодействия. (б) — модель, соответствующая марковскому процессу принятия решения без иерархической декомпозиции. (с) — модель агента с умениями	118
2.12	Среда Crafter (слева) и список подзадач, которые выделяются в этой среде (справа).	126
2.13	Архитектура NSLP с выделенным концептуальным уровнем, на котором используются языковые модели для генерации гипотезы и составления символьного плана действий.	127
2.14	Общая архитектура использования больших языковых моделей (БЯМ) для задачи построения плана поведения воплощенного агента.	133

3.1	Архитектура NSLP с выделенным сенсорным уровнем, на котором используется подход «актора-критика» для обновления латентной модели среды и плоской стратегии агента без использования навыков.	145
3.2	Схема расчета полезности действий в алгоритме TreeQN с глубиной $d = 2$	147
3.3	Наблюдение в среде Crafter. В центре расположен игровой персонаж, внизу размещены инвентарь. В поле зрения находятся различные типы местности (трава, вода) и деревья. Агент может съесть корову, которая движется справа, для пополнения уровня пищи.	151
3.4	График зависимости скользящего среднего суммарного вознаграждения за эпизод от количества шагов в среде Atari Pong. Произведено усреднение кривых по трем запускам алгоритмов. Затенение показывает минимальное и максимальное значение в группе запусков.	152
3.5	Графики зависимости метрики score от количества шагов в среде Crafter для всех рассматриваемых алгоритмов с бюджетом взаимодействия со средой не более 4.5 млн шагов. Для оси ординат применен логарифмический масштаб. Произведено усреднение кривых по трем запускам алгоритмов. Затенение показывает минимальное и максимальное значение в группе запусков.	153
3.6	Графики зависимости метрики score от количества шагов в среде Crafter для медленно обучающихся алгоритмов с бюджетом взаимодействия со средой не более 120 млн шагов. Произведено усреднение кривых по трем запускам алгоритмов. Затенение показывает минимальное и максимальное значение в группе запусков.	153
3.7	Графики зависимости скользящего среднего суммарного вознаграждения за эпизод от количества шагов в среде Crafter для всех рассматриваемых алгоритмов с бюджетом взаимодействия со средой не более 4.5 млн шагов. Произведено усреднение кривых по трем запускам алгоритмов. Затенение показывает минимальное и максимальное значение в группе запусков.	154
3.8	Графики зависимости скользящего среднего суммарного вознаграждения за эпизод от количества шагов в среде Crafter для медленно обучающихся алгоритмов с бюджетом взаимодействия со средой не более 120 млн шагов. Произведено усреднение кривых по трем запускам алгоритмов. Затенение показывает минимальное и максимальное значение в группе запусков.	154
3.9	Общий алгоритм оптимизации стратегии с помощью модели.	158
3.10	Схема генерации выборок (семплирования) в процессе обучения. Здесь buffer — это память предшествующих	159
3.11	Демонстрация распространения полезности из граничных состояний в алгоритме SAC с накоплением ошибок. Среда — это одномерный отрезок, по которому может перемещаться агент. Самое далёкое от начала состояние даёт максимальное вознаграждение и достигается за 3 шага. Среднее значение функции полезности состояния резко возрастает, когда максимальное разрешенное количество шагов (l) равно 3.	160

3.12 Оценка метода в среде <i>Reacher</i> . (а) Средняя ошибка предсказания углов сочленений падает ниже 1 градуса на горизонте предсказания в 4 шага. (б) Предлагаемый метод обеспечивает более высокую эффективность выборок среди <i>Reacher</i> , требуя в 3 раза меньше элементов выборки, чем SAC.	161
3.13 В среде <i>Cheetah-run</i> рассматриваемый метод демонстрирует более высокую эффективность выборок из среды. Однако, при наличии 2×10^6 шагов Dreamer-v3 показывает более высокое качество по сравнению с предлагаемым методом. Все сравниваемые подходы используют идентичные векторные наблюдения.	162
3.14 (слева) Предлагаемый метод демонстрирует более высокое качество в среде <i>DoorOpen</i> по сравнению с Dreamer-v3 с размером модели XL (xlarge) и гиперпараметрами по умолчанию. Примечательно, что алгоритм Dreamer-v3 не решает задачу открытия двери. (справа) Кадр из среды <i>DoorOpen</i>	162
3.15 Архитектура NSLP с выделенными блоками интеграции обучения стратегии и планирования по объектной модели мира M_O	163
3.16 Подход MAMCTS адаптирует алгоритм MCTS для многоагентных задач, рассматривая варианты выбора действий для каждого агента как отдельные узлы в дереве, тем самым уменьшая коэффициент ветвления. На рисунке показаны все четыре этапа MCTS: а) выбор (selection); б) расширение (expansion); в) симуляция (simulation); и г) обратное распространение (backpropagation) для многоагентной постановки задачи. Действия, ведущие к вершинам, показанным пунктирными линиями, не рассматриваются на этапе выбора. Заполненные круги представляют агентов, в то время как пустые круги представляют соответствующие им цели.	170
3.17 Примеры карт, использованных во время обучения: (а) Кооперативные случайные карты, которые были выбраны в соответствии с их сложностью. (б) Карты лабиринтов, сгенерированные с помощью пакета labmaze. Агенты представлены заполненными кругами, а их цели — пустыми кругами. У каждого агента есть своя цель.	172
3.18 На рисунке изображена схема алгоритма COSTTRACER. Подход использует две матрицы в качестве входных данных: одна кодирует препятствия, нормализованные обратные затраты на выполнение; другая содержит позиции локальных агентов. Весь конвейер обучается с помощью алгоритма PPO, используя функцию вознаграждения, которая обеспечивает положительную обратную связь только тогда, когда агент приближается к своей глобальной цели.	177

3.19 Схема подхода MATS-LP. Сначала строится внутренний МППР (IMDP) с использованием глобальной карты статических препятствий, агентов в наблюдении и их текущих целей. Этот МППР служит основой для планирования с использованием MCTS. При таком подходе каждый узел дерева представляет совместное действие всех агентов, присутствующих в IMDP, вместе со статистикой связанных узлов. Вероятности действий и полезности узлов вычисляются с использованием алгоритма COSTTRACER. Необходимо отметить, что полезность для каждого агента вычисляется индивидуально, а оценка узла выводится как сумма полезностей по всем агентам. Процедура планирования концентрируется исключительно на агентах, находящихся в непосредственной близости. Например, в данном сценарии это включает в себя самого агента (выделен красным цветом) и коричневого агента. Это решение ограничить рассмотрение ближайших агентов значительно упрощает процесс принятия решений. Действия агентов визуально представлены с помощью круговых иконок направлений, а жирные стрелки выделяют действия с наибольшей вероятностью. Для удаленных агентов рассматривается только одно действие с максимальной вероятностью.	179
3.20 Средняя пропускная способность (throughput) MATS-LP, SCRIMP и PRIMAL2 на случайных картах 20×20 с различной плотностью препятствий. Символом \star отмечены подходы, которые были обучены на картах соответствующего типа. Заштрихованные области указывают на 95% доверительные интервалы.	182
3.21 Средняя пропускная способность MATS-LP, SCRIMP и PRIMAL2 на лабиринтоподобных картах различных размеров. Символом \star отмечены подходы, которые были обучены на картах соответствующего типа.	183
3.22 Средняя пропускная способность и среднее время принятия решения MATS-LP, SCRIMP и PRIMAL2, а также исследование влияния компонент алгоритма MATS-LP на складской карте. Заштрихованные области указывают на 95% доверительные интервалы.	184
3.23 Возможные способы связи состояний и действий в модели. S^n — контекст из n состояний. Z — латентное пространство представлений. M^k — ансамбль из k отдельных моделей. A^h — последовательность из h действий. \tilde{S}, \tilde{Z} — состояние и представление, в котором окажется агент через h шагов от начального.	187
3.24 Уровни внутренне мотивированного агента. На каждом из уровней методы внутренней мотивации предлагают свой аналог: стратегии, вознаграждения или цели.	189
3.25 Сбор данных для обучения. Целевые и исследовательские стратегии набирают данные для обучения из модели мира и среды в память D_g, D_ϵ, D_M	189
3.26 Слева: методы внутренней мотивации явным образом модифицируют вознаграждение агента. Справа: методы внутренней мотивации реализуют исследовательскую стратегию, явно корректируя стратегию агента.	195

3.27 Модель мира явно определяет цели исходя из морфологии среды и неявно через определение исследовательской стратегии, достигнутые которой состояния становятся целями.	196
4.1 Общая архитектура предлагаемой модели VQ-SA.	201
4.2 Манипулирование отдельными объектами в сцене.	204
4.3 Слева: результаты подсчета метрики DQCF-micro. Справа: результаты подсчета метрики DQCF-macro.	206
4.4 Архитектура NSLP с выделенным нейросимвольным уровнем, на котором используется слотовый подход для построения объектного представления наблюдений агента.	207
4.5 Сравнение архитектур предлагаемого модуля смеси слотов (SMM) и модуля слотового внимания. Зеленый цвет используется для обозначения шагов, задействованных в обоих модулях. SMM включает оценку центров кластеров (μ), расстояния между центрами кластеров и отнесенными к ним векторами (σ , оранжевые шаги) и веса априорной смеси (π , красные шаги). Объединение μ и σ служит итоговым представлением слотов, а π может использоваться для идентификации пустых слотов, которые не содержат информации о каких-либо объектах, $f(\mathbf{x}, \mu, \sigma)$ является логарифмом гауссовой функции плотности. Модуль слотового внимания (SA) оценивает только центры кластеров.	215
4.6 Примеры генерации изображений с использованием Image GPT, обусловленные представлениями различных слотов. Изображения с синими границами взяты из модели с модулем слотового внимания, а изображения с зелеными границами сгенерированы с использованием слотов из модуля смеси слотов. Красный цвет границы обозначает исходные изображения.	218
4.7 Примеры всех качественно некорректно сгенерированных изображений из случайного пакета размером 64 примеров. В шести случаях реконструкция с использованием слотового внимания привела к неправильному порядку объектов (синий круг) или потере одного объекта (красный круг). В оставшихся двух примерах оба модуля имеют неправильную реконструкцию (зеленый круг).	218
4.8 Пример обнаружения объекта в наборе данных ClevrTex. Первый столбец представляет исходные истинные изображения, второй — реконструкции широковещательного декодировщика. Последующие столбцы соответствуют слотам масок внимания.	220
4.9 Пример редактирования изображений из набора данных Shape Stacks и Bitmoji с использованием выборки концептов. Этот метод позволяет точно редактировать отдельные концепты: в первой строке показана возможность изменения формы, цвета отдельных объектов или манипуляция с фоном. Во второй строке приведен пример создания похожих, но немного отличающихся причесок с помощью выборки концептов.	222

4.10 Архитектура NSLP с выделенным нейросимвольным уровнем, на котором формируется объектное представление для обучения стратегии агента с использованием модели среды.	223
4.11 Схема работы предлагаемого подхода ROCA, в котором модель формирует стратегию, извлекая объектно-центричные представления из исходного изображения и обрабатывая их в виде полного графа.	227
4.12 Схема взаимосвязи компонент ROCA. Метод состоит из предварительно обученной модели SLATE с зафиксированными весами, которая извлекает объектно-центрические представления из наблюдения в виде изображений, и модулей на основе GNN: модели перехода, модели вознаграждения, модели полезности состояния и модели актора. Модели перехода и вознаграждения формируют модель мира. Модель мира и модель полезности состояния вместе составляют модуль критика, который предсказывает Q-значения.	228
4.13 Показатель отдачи и успешности решения задачи усредненное по более чем 30 эпизодам и трем запускам эксперимента для моделей ROCA, DreamerV3 и OCRL. ROCA учится быстрее или достигает более высоких показателей, чем базовые алгоритмы. Заштрихованные области указывают на стандартное отклонение.	233
4.14 Среднее значение отдачи, усредненное по 30 эпизодам и трем запускам эксперимента, для моделей ROCA, DreamerV3 и OCRL в задаче Navigation10x10. ROCA демонстрирует лучшую производительность, чем базовые алгоритмы, но все равно не решает задачу. Заштрихованные области указывают на стандартное отклонение.	234
4.15 Исследование влияния компонент. SAC-CNN — версия SAC со стандартным сверточным кодировщиком. SAC-SLATE — версия SAC с предварительно обученным кодировщиком, который усредняет представления объектов для получения итогового представления текущего состояния. SAC-WM-SLATE — модификация SAC-SLATE, которая использует монолитную модель мира в своем критике. SAC-GNN-SLATE — объектно-центрическая версия SAC с предварительно обученным кодировщиком, который использует GNN в акторе и критике. ROCA (без настройки) — версия ROCA без настройки целевой энтропии. ROCA (без действий в ребрах GNN) — версия ROCA, в которой функции ребер не принимают действия в качестве входных данных. ROCA превосходит рассмотренные базовые алгоритмы. Заштрихованные области указывают на стандартное отклонение.	235
4.16 Архитектура NSLP с выделенным основным циклом генерации плана действий с использованием языковой модели и их выполнения на основе библиотеки предобученных навыков.	237
4.17 Задача коллaborативного взаимодействия в среде агента и пользователя включает в себя следующие этапы: пользователь предоставляет инструкции агенту, а агент создает фигуру в среде на основе этих инструкций.	239

4.18 Схема предлагаемого метода построения подзадач по языковой инструкции (L2S). Агент состоит из трех частей. Сначала языковой модуль преобразует инструкции на естественном языке в текстовую последовательность подзадач. Затем менеджер задач преобразует эту последовательность в упорядоченный список подзадач. Наконец, модуль стратегии выполняет предварительно обученную стратегию для выполнения каждой подцели, одновременно отслеживая успешность действий.	243
5.1 Используемая в экспериментах робототехническая система.	254
5.2 Взаимодействие программных модулей системы в рамках предлагаемой архитектуры STRL-Robotics.	256
5.3 Структурная схема процесса реконструкции трехмерной карты местности.	257
5.4 Пример построенной лидарной карты этажа офисного здания (пятый этаж корпуса Физтех.Цифра МФТИ). Для лучшей визуализации показаны только угловые точки. Слева — вся карта, справа — увеличенная часть карты.	257
5.5 Демонстрация планирования с увеличенными препятствиями. На рисунке справа разные уровни увеличения препятствия показаны разными оттенками серого. Слева — геометрические размеры агента, используемые для увеличения препятствий, справа — спланированная траектория при условия нахождения робота в запрещенной области. Красным цветом отмечен геометрический путь, синим — примерное начало траектории движения робота.	260
5.6 Работа алгоритма следования пути. Слева — исходная версия, справа — модифицированная версия.	261
5.7 Работа алгоритмов следования по траектории. Слева — алгоритм из работы [260], справа — модифицированная версия. Здесь R_1, R_2 и P_1, P_2 — фактическое и желаемое положения робота в моменты t_1 и t_2 соответственно, $t_1 < t_2$	262
5.8 Десять последовательных путей, использованных для экспериментов (разные цвета использованы для лучшей различимости. S и F — точки старта и финиша).	267
5.9 Координация ROS-узлов при вызове лифта.	270
5.10 Последовательность движений при вызове лифта.	271
5.11 Архитектура NSLP с выделенным циклом генерации действия по наблюдению и библиотекой навыков, включающей как обученные, так и классические стратегии.	272
5.12 Схема процесса принятия решений SkillFusion при решении задачи навигации к объектам, используемого на реальном роботе. Процесс состоит из классической части и части, основанной на обучении с подкреплением. Каждая часть обладает навыками исследования среды и достижения цели. Чтобы выбрать подходящий навык в каждый момент, реализуется решатель (decider) слияния навыков, который выбирает действие из набора навыков с наибольшей полезностью.	273
5.13 Пример объединения по максимуму построенной с помощью SLAM карты. Серым цветом обозначена неизвестная область, черным — клетки с препятствиями, а белым — клетки свободного пространства.	280

5.14 Предлагаемая архитектура нейросетевого аппроксиматора для совместного обучения навигационных навыков.	282
5.15 Пример процесса выбора навыков в рамках одного эпизода. Синий цвет траектории обозначает выполнение обучаемого навыка исследования среды, темно-синий обозначает обучаемый навык достижения цели, а красный цвет обозначает классические навыки.	285
5.16 Примеры работы верхнеуровневой стратегии в симуляционной среде. Красная линия — это траектория, выполненная с классическим навыком, светло-синяя линия — с обучаемым навыком исследования среды, а темно-синяя — с обучаемым навыком достижения цели.	286
5.17 Сравнение кривых обучения навыка достижения цели с использованием зафиксированного кодировщика на базе модели CLIP и с обучаемым кодировщиком на базе модели ResNet.	288
5.18 Роботизированная платформа на базе Clearpath Husky с камерой ZED (слева). ПолYGON, который был использован для оценки результатов в реальных условиях (справа).	290
5.19 Траектории движения реального робота с разными подходами: обучаемым (вверху), классическим (посередине) и полным методом SkillFusion (внизу). Красный прямоугольник обозначает целевой объект, красный круг — начальную точку, а белый круг с синей стрелкой — конечную точку робота и его ориентацию.	291
5.20 Схема использования NSLP агента в качестве модуля в системе управления Apollo.	293
5.21 Пример реализации сценария обгона динамических препятствий в системе управления Apollo с использованием предложенной концепции адаптивного планирования. Зеленые тонкие линии — предсказываемые траектории объектов, голубая полоса — планируемая траектория агента.	293
5.22 Элементы вектора наблюдения для локальной обучаемой стратегии.	297
5.23 Этапы программы обучения стратегии агента.	299
5.24 Архитектура нейросетевых аппроксиматоров актора и критика.	300
5.25 Модули восприятия, определения границы сцены, генерации траектории POLAMP в виде топиков CyberRT.	300
5.26 Результаты обучения алгоритма POLAMP в задаче генерации маневра парковки.	302
5.27 Различные сценарии парковки с динамическими препятствиями были протестированы в симуляторе SVL (рисунки слева) и в Apollo (рисунки справа).	302
6.1 Концептуальная схема работы когнитивного семиотического агента. Обозначения расшифровываются в основном тексте.	306
6.2 Схема функций связывания Ψ_p^m , $\Psi_m^a \Psi_a^p$. С тильдой обозначены узлы каузальной сети до связывания в знак. Функции именования компонент: \mathfrak{P} (для образа), \mathfrak{M} (для значения), \mathfrak{A} (для смысла).	309
6.3 Схема обучения акторной каузальной сети с помощью алгоритма иерархического актор-критика.	317

6.4	Пример расположения рецептивного поля агента и схемы кодирования признаков для функции распознавания Φ^p	322
6.5	Получение вознаграждения семиотическим агентом с использованием сценариев реакции среды (по горизонтальной оси — кол-во шагов, по вертикальной — скользящее среднее по пяти шагам).	323
6.6	Примеры из «Scale B 16 PF» опросника R.Cattell и их модификация для запроса БЯМ.	329
6.7	Примеры из методики Рубинштейн «Выделение существенных признаков» и их модификация для запросов к БЯМ.	330

Список таблиц

1	Сравнения алгоритмов для задач воплощенного искусственного интеллекта на основе предобученных БЯМ.	132
2	Результаты БЯМ в трех различных режимах планирования с запросами для конкретной задачи.	140
3	Результаты БЯМ в трех различных режимах планирования со смешанным запросом.	141
4	Производительность различных алгоритмов сравнивалась на 100 совместных случайных картах размером 16×16	173
5	Результаты тестирования алгоритмов на 100 лабораторных картах лабиринтов размером 15×15 . Алгоритм подцели МАМCTS показал наилучшие результаты. . .	173
6	Параметры алгоритмов COSTTRACER и MATS-LP. В столбце «Tuned» указаны параметры, которые были оптимизированы с помощью гиперпараметрического поиска.	186
7	Характеристика рассматриваемых методов внутренней мотивации по наличию соответствующих компонент на разных уровнях агента (вознаграждения, стратегии, цели, см. рисунок 3.24) и типу сигнала внутренней мотивации.	190
8	Результаты работы на задаче прогнозирования множества свойств объектов на наборе данных CLEVR.	204
9	Результаты исследования влияния компонент модели. В скобках указано общее количество векторных представлений.	205
10	Качество реконструкции (среднее значение \pm std для четырех запусков). SLATE(SA) — оригинальная модель SLATE, SLATE(SMM) — модифицированный SLATE с SMM вместо SA. В качестве верхней оценки метрики приведены результаты FID для dVAE.	217
11	Качество работы в задаче прогнозирования множества свойств для набора данных CLEVR (AP в %, среднее значение \pm std для четырех запусков в экспериментах). Для слотового внимания [471] и для iDSPN [659] результаты взяты из оригинальных статей. SA* — это модель слотового внимания, обученная с теми же условиями, что и SMM: отсоединенные слоты на последней итерации с масштабированием координат до диапазона [-1, 1]. Результаты SA-MESH взяты из оригинальной статьи [630].	219
12	Значения метрики FG-ARI для набора данных ClevrTex.	220
13	Средняя точность (среднее значение \pm std для четырех запусков эксперимента) с различными пороговыми значениями расстояния для задачи прогнозирования множества свойств в наборе данных CLEVR после 100 тысяч итераций обучения.	222

14	Сравнение метрики F1 базовой модели FlanT5 при обучении с использованием двух разных методов аугментации данных. Метод А предполагает замену цветов, в то время как метод ABC включает как замену цветов, так и вращение фигур. Наборы данных A(strt) и ABC(strt) были использованы для выборки эпох обучения путем стратификации. В столбце «data» представлены результаты для модели, обученной на исходном наборе данных IGLU, в то время как в столбце «prim» показаны результаты для модели, обученной на исходном наборе данных, где каждая инструкция преобразуется с использованием метода декомпозиции фигуры на примитивы.	248
15	Средние значения метрик по всем тестовым заданиям в наборе данных IGLU по пяти запускам экспериментов. Алгоритмы T5 (<i>770 миллионов параметров</i>) и Pegasus (<i>568 миллионов параметров</i>) — это реализации агентов из общедоступного решения соревнования IGLU. BrainAgent — это сквозная модель с 640 миллионами параметров без декомпозиции метода на языковую модель и модель стратегии. L2S работает с <i>220 миллионами параметров</i> на базе Flan-T5. Лучший подход выделен полужирным , а лучшая модель, включая проприетарные реализации, такие как Chat-GPT, обозначена розовым цветом . . .	249
16	Результаты для алгоритма следования по пути.	267
17	Результаты для предложенного алгоритма слежения.	267
18	Сравнение моделей сегментации объектов на тестовой выборке.	268
19	Ошибка позиционирования кнопки, м.	269
20	Метрики качества метода SkillFusion по сравнению с базовыми алгоритмами на наборе данных HM3D.	287
21	Исследование влияния различных навыков на итоговую эффективность метода SkillFusion в рамках одного эпизода.	289
22	Исследование влияния модуля семантической сегментации на метрики метода SkillFusion.	289
23	Результаты отдельных тестов классического и основанного на обучении подходов на реальном роботе в сравнении с полным методом SkillFusion.	291

Приложение А

Акты о внедрении и использовании результатов исследования

	<p>ООО «Фаст Сенс Студия», Фактический адрес: 121509, г. Москва, ул.Бережковская набережная, 20с6, этаж 2 ОГРН: 1177746494254 ИНН: 9710029138 КПП: 771401001</p>
УТВЕРЖДАЮ Директор ООО «Фаст Сенс Студия» Румянцев Ю. А.	
24 мая 2024 г.	
АКТ о внедрении результатов диссертационной работы Панова Александра Игоревича	
на тему «Методы и алгоритмы нейросимвольного обучения и планирования поведения когнитивных агентов», представленной на соискание ученой степени доктора физико-математических наук	
Комиссия в составе: - председателя: Генерального директора Румянцева Юрия Андреевича - членов: Исполнительного директора Бирюкова Павла Михайловича	
составили настоящий акт о том, что результаты, полученные в диссертационной работе Панова А.И., а именно:	
1) Метод и алгоритм интеграции планирования и обучения с подкреплением, применимый для многоагентной постановки задач в динамических средах. 2) Архитектурная реализация интеграции тактического уровня управления с построением сенсорного описания ситуации (динамической карты проходимости) и стратегического уровня по планированию действий перемещения наземного робота.	
использованы при выполнении НИОКР по теме «Разработка математического обеспечения для решения задач построения динамической карты проходимости и планирования на ее основе движения мобильных наземных роботов» в части реализации методов планирования движения мобильных наземных роботов по динамической карте проходимости.	
Председатель комиссии: Члены комиссии:	
	
Румянцев Ю. А. Бирюков П. М.	

АКЦИОНЕРНОЕ ОБЩЕСТВО
**«НАУЧНО-ПРОИЗВОДСТВЕННЫЙ КОМПЛЕКС
 «БОРТОВЫЕ ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ»**



УТВЕРЖДАЮ
 Генеральный директор АО «НПК «БИС»

O.В. Максёнков

06.2024 г.

АКТ

об использовании результатов диссертационной работы

Панова Александра Игоревича

г. Москва

20.06.2024

на тему «Методы и алгоритмы нейросимвольного обучения и планирования
 поведения когнитивных агентов», представленной на соискание ученой степени
 доктора физико-математических наук

Результаты диссертационной работы «Методы и алгоритмы нейросимвольного обучения и планирования поведения когнитивных агентов» обладают высокой актуальностью и представляют практический интерес для решения различных робототехнических задач и задач автоматизации производственных процессов.

В рамках выполнения научно-исследовательской работы «Исследование и разработка математического обеспечения для решения задач автоматического планирования маршрутов и траекторий движения, функционирующего в составе комплекта аппаратуры управления транспортного средства» (договор № Ю-51/2021 от 25 августа 2021г.) по заказу Акционерного общества «Научно-производственный комплекс «Бортовые интеллектуальные системы» был разработан адаптивный метод планирования маневров, интегрирующий методы обучения с подкреплением и эвристического

планирования пути для задач обгона и парковки. Полученные в рамках исследовательской работы результаты привели к улучшению качественных характеристик (быстродействие, достижение заданных критериев оптимизации) существующих методов решения задачи планирования траектории движения транспортного средства с учетом наличия дискретной карты проходимости, заданных габаритов и кинематических ограничений транспортного средства, требований к ориентации транспортного средства в конечной точке, а также информации о параметрах движения динамических препятствий.

Научный руководитель



И.И. Итенберг

ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
«ИНТЕГРАЦИЯ НОВЫХ ТЕХНОЛОГИЙ»

АКТ

об использовании результатов диссертационной работы

Панова Александра Игоревича

г. Москва

20.06.2024

на тему «Методы и алгоритмы нейросимвольного обучения и планирования
поведения когнитивных агентов», представленной на соискание ученой степени
доктора физико-математических наук

Результаты диссертационной работы «Методы и алгоритмы нейросимвольного
обучения и планирования поведения когнитивных агентов» обладают высокой
актуальностью и представляют практический интерес для решения различных задач в
беспилотных технологиях.

В диссертационной работе Панова А.И. был разработан метод адаптивного
планирования маневров беспилотного автомобиля в задачах парковки, перестройки и
обгона динамических препятствий, а также созданы элементы программно-
алгоритмического инструментария, основанного, в том числе, на полученных
теоретических результатах диссертации в областях обучения с подкреплением и
планирования пути. Данный инструментарий служит для решения задачи генерации
действий робототехнической платформы (беспилотного автомобиля) в сложной
динамической среде и позволяет использовать в системе управления для генерации
действий как обучаемые, так и классические планировочные компоненты.

Полученные результаты нашли свое практическое применение в научно-

исследовательской работе под названием «Разработка математического обеспечения для решения задач автоматического управления движением автомобильного транспортного средства в условиях дорог общего пользования» (договор № 1-2022/НИР от 15 июня 2022г.), выполненной для ООО «ИнтеграНТ». В свою очередь, в результате выполнения научно-исследовательской работы был разработан комплект экспериментальных программных реализаций алгоритмов автоматического управления беспилотным автотранспортным средством (АТС) в условиях дорог общего пользования, с учетом кинематических и динамических параметров шасси АТС, информации о дорожной инфраструктуре, получаемой из HD карт в формате OpenDrive, заданных скоростных ограничений.

Благодаря использованию экспериментальных программных реализаций, на данных с АТС были достигнуты следующие метрики качества при проезде перекрестков: средняя длина траекторий 44м, время проезда – 27 сек., метрика AOL – 1,13 рад/м, что улучшает качество по сравнению с ручным управлением на 10%.

Генеральный директор

T.B. Михайлова



УТВЕРЖДАЮ
Старший управляющий директор
директор Центра робототехники
ПАО «Сбербанк»
А.С. Гончаренко

07.06.2024



АКТ

об использовании (внедрении) результатов диссертационной работы

Панова Александра Игоревича

на тему «Методы и алгоритмы нейросимвольного обучения и планирования

поведения когнитивных агентов», представленной на соискание ученой степени доктора

физико-математических наук

Результаты диссертационной работы «Методы и алгоритмы нейросимвольного обучения и планирования поведения когнитивных агентов» обладают высокой актуальностью и представляют практический интерес для решения различных задач мобильной и антропоморфной робототехники.

Методы планирования с помощью больших языковых моделей, разработанные Пановым А.И., позволяют улучшить качество и строить более эффективные архитектуры управления робототехническими платформами различного назначения, используемыми в Центре робототехники ПАО «Сбербанк» (далее – Центр).

Данные работы выполнены в рамках договора НИР(ОКР), заключенного между ПАО «Сбербанк» и МФТИ, где Панов А.И. является Директором Научно-образовательного центра когнитивного моделирования. Работы выполнены в рамках проекта «Управление действиями робота. Разработка системы управления роботом с модулем планирования по языковым инструкциям для сортировки объектов в помещении с использованием мобильного робота и манипулятора» (Приложение № 3 к Дополнительному соглашению № 4 от 21 сентября 2023 года к Договору с Индустриальным партнером № 0826/СБЕР-МФТИ от 26 августа 2021 г.).

Центр ведет работы по объединению робототехники и искусственного интеллекта для создания умных роботов нового поколения. Разрабатывает технологии, которые помогут роботам выполнять действия наравне с человеком, выполнять определенные повседневные задачи, взаимодействуя с реальным миром на основе собственного опыта.

Старший управляющий директор
директор Центра робототехники
ПАО «Сбербанк»



Гончаренко А.С.