RL в многоагентном планировании: SCRIMP и DCC

Андрей Яременко
СПбГУ, МКН
andreiyaremenko9@gmail.com

Николай Степанов
СПбГУ, МКН
elteammate@gmail.com

Проектная работа в рамках курса
«Методы и алгоритмы эвристического поиска»
Константин Сергеевич Яковлев, 2025

Введение в многоагентное планирование

Рис. 1. Роботы в Amazon (оранжевые) переносят складские стелажи. Многоагентное планирование занимается организацией их совместной работы.

Многоагентное планирование (multi-agent planning) представляет собой одну из ключевых областей исследований в искусственном интеллекте. Оно направлено на координацию действий нескольких агентов для достижения общей цели, обычно построения маршрута до точек назначения. Такие системы находят широкое применение в множестве областей.

Например, в логистике многоагентное планирование позволяет оптимизировать маршруты доставки, в робототехнике оно используется для координации работы групп роботов, например, в складских системах, а в игровой индустрии такие системы применяются для создания сложных сценариев взаимодействия между персонажами.

Несмотря на значительные достижения в этой области, многоагентное планирование остаётся сложной задачей из-за высокой размерности пространства состояний, неопределённости среды и конфликтов между интересами агентов. В последние годы одним из наиболее перспективных подходов к её решению стал метод обучения с подкреплением (reinforcement learning, RL), который позволяет агентам адаптироваться и учиться на основе опыта взаимодействия со средой. Отдельно выделяется RL с механизмами коммуникации между агентами. В данной статье рассматриваются и сравниваются два таковых: **SCRIMP** [1] и **DCC** [2].

Задача многоагентного планирования

В широком смысле задача многоагентного планирования заключается в том, чтобы разработать стратегии для группы агентов, которые действуют в общей среде. Под агентами понимаются сущности, способные автономно или централизовано принимать решения. Среда, в которой действуют агенты, может быть динамической, неопределённой и содержать множество ограничений, таких как ресурсные ограничения или конфликты между целями агентов.

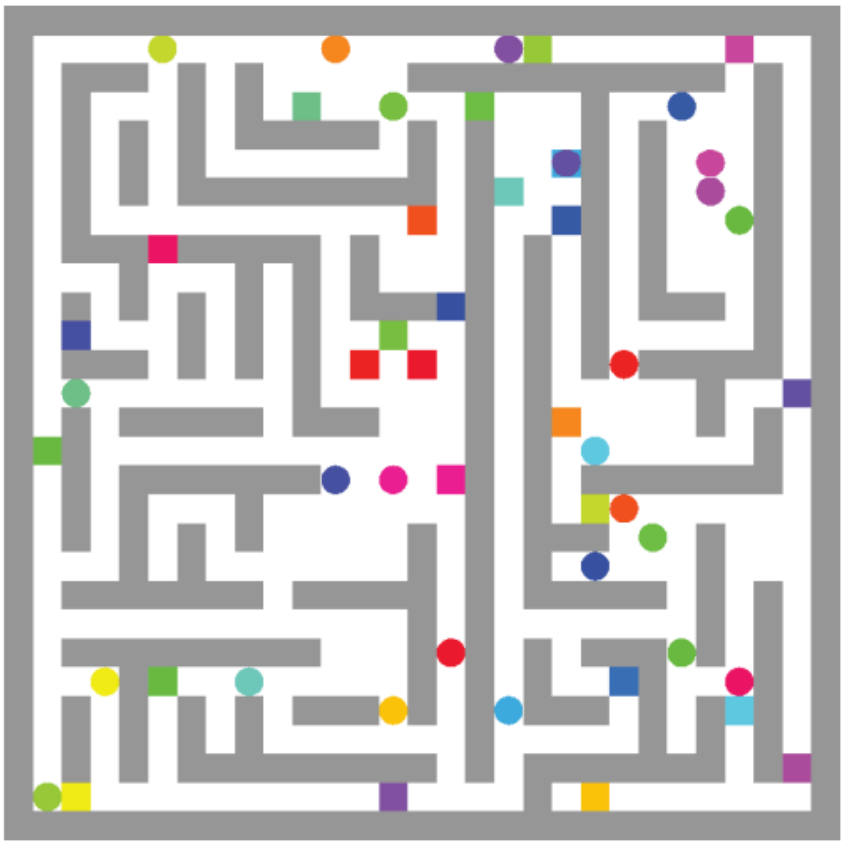

Рис. 2. Пример задачи. Препятствия обозначены серым, разноцветные квадраты — агенты в начальном положении, а круги соответствующих цветов — их конечные положения.

Мы ограничиваем задачу, рассматривая ее в контексте прямоугольных сеток с препятствиями, где агенты должны перемещаться от начальных позиций к конечным, избегая столкновений друг с другом и с препятствиями. Под сеткой подразумевается дискретное представление среды в виде *клеток* или *вершин*, каждая из которых может быть проходимой или препятствием. Соседние по сторонам клетки связаны *ребрами*, агенты могут по ним передвигаться. Время в задаче *дискретно*: каждый момент времени агент принимает решение, в какую соседнюю клетку перейти или остаться на месте, и совершает его. *Путь агента* — последовательность клеток, которые агент посещает, начиная с начальной позиции и заканчивая целевой. Для каждого агента известны его начальные и конечные позиции. Ситуация, в которой два или более агента пытаются одновременно занять одну и ту же клетку или пересечься на пути, называется *конфликтом*. Задача состоит в поиске таких путей, что по ним агенты из своих начальных позиций достигают конечных, без конфликтов.

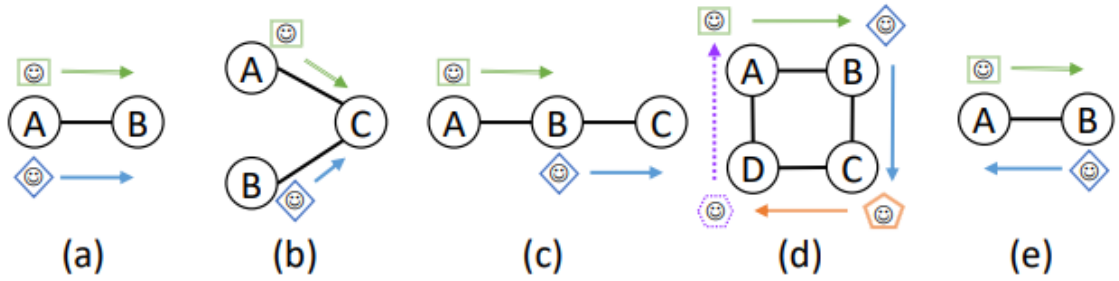

Рис. 3. Различные типы конфликтов в многоагентном планировании [3]. В нашей постановке конфликтами являются ситуации (a), (b) и (e).

Формальная постановка задачи описана в [3]. В нашем случае, она формулируется так.

Для задачи с k агентами, пусть дана тройка
- $G = \langle V, E \rangle$ — неориентированный граф, где $V \subset \mathbb{Z}^2$, и евклидово расстояние между инцидентными вершинами равно 1,
- $s : \{1, 2, ..., k\} \longrightarrow V$ — *начальная позиция* каждого агента,
- $t : \{1, 2, ..., k\} \longrightarrow V$ — *конечная позиция*, или *цель* каждого агента.

Назовем *действием* или *шагом* функцию $a : V \longrightarrow V$, вида $a(v) = v$ (агент остается на месте, stay action), или $a(v) = v'$, где $(v, v') \in E$ (агент передвигается по ребру).

Обозначим $\pi_i = \left(a^i_1, ..., a^i_{n_i}\right)$ — последовательность действий, совершаемых агентом i, и $\pi_i[x] = (a^i_x \circ a^i_{x-1} \circ ... \circ a^i_1)(s(i))$ — положение агента i в момент x. Если $x > |\pi_i|$, положим $\pi_i[x] = \pi_i(|\pi_i|)$. Последовательность действий π_i называется *путем* если $\pi_i[|\pi_i|] = t(i)$. *Решением* называется k путей для каждого агента.

Пусть π_i, π_j — пара путей. *Вершинным конфликтом* называется ситуация, когда существует x для которого $\pi_i[x] = \pi_j[x]$. *Конфликтом обмена* (swapping conflict) называется ситуация, когда существует x для которого $\pi_i[x] = \pi_j[x + 1]$ и $\pi_j[x] = \pi_i[x + 1]$.

Решение называется *корректным*, если для любой пары путей различных агентов, между этими путями нет конфликтов любого рода. Задача многоагентого планирования (MAPF) есть задача поиска корректного решения, обычно минимизирующая некоторую метрику качества. Частыми метриками является:
- Makespan: количество действий, необходимых, чтобы все агенты дошли до целей, или $\max_i |\pi_i|$.
- Flowtime: общее количество шагов в решении: $\sum_i |\pi_i|$.

Далее под «решениями» будут подразумеваться только корректные решения. Иногда задачи мы будем называть *картами*.

Подходы к решению

Применительно к MAPF не существует «серебрянной пули». Все методы имеют достоинства и недостатки [4].

Методы поиска оптимальных решений (например поиск в пространстве состояний) обычно имеют большую вычислительную сложность из-за экспоненциального роста размера пространства состояний. Быстрые алгоритмы, например приоритезированное планирование, значительно выигрывают в производительности, но решения могут быть далеки от оптимальных. Методы на основе сведения к другим NP-полным задачам также страдают от низкой производительности.

Компромиссом между быстрыми и точными методами могут являться децентрализованные подходы. В них агенты принимают решения автономно, на основе своего окружения и сообщений от других агентов. Такие подходы масштабируются линейно с числом агентов. Качество решения тогда зависит от качества реализации отдельного агента, и качества механизма взаимодействия агента со средой. Роль последнего заключается в решении проблем подхода, например, разрешения конфликтов. Если в централизованных методах конфликты означают некорректность реализации, то в децентрализованных подходах конфликты абсолютно нормальны, и, более того, ожидаемы. Необходим механизм для борьбы с ними.

Одной из реализаций децентрализованного планирования является PRIMAL [5]. Каждый агент моделируется рекуррентной нейросетью, которая принимает на вход окружение агента, и позицию цели, и возвращает вероятности агента совершить одно из 5 возможных действий.

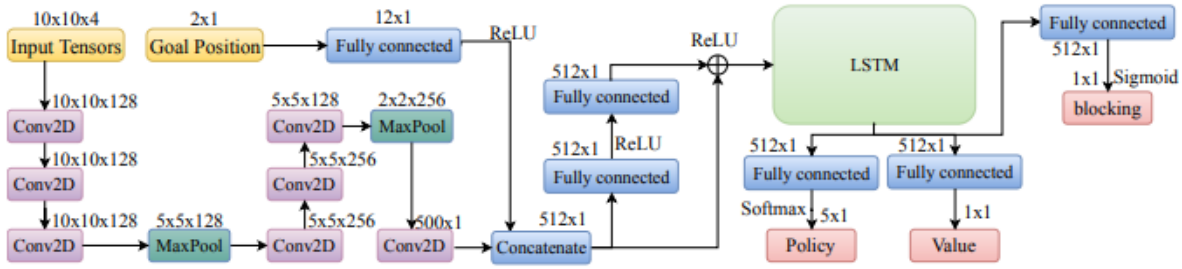

Рис. 4. Архитектура сети в PRIMAL [5].

Она обучается с помощью обучения с подкреплением и обучения имитации, подобно [6]. За счет того, что поведение всех агентов одинаково и зависит исключительно от окружения, а обучение минимизирует суммарную награду, в поведении агентов наблюдается кооперирование.

В такой модели существуют векторы улучшения. Самое очевидное, добавить возможность агентам общаться друг с другом. Именно этим выделяются рассматриваемые нами модели.

SCRIMP

SCRIMP, или Scalable Communication for Reinforcement- and Imitation-Learning-Based Multi-Agent Pathfinding [1] предлагает следующую архитектуру.

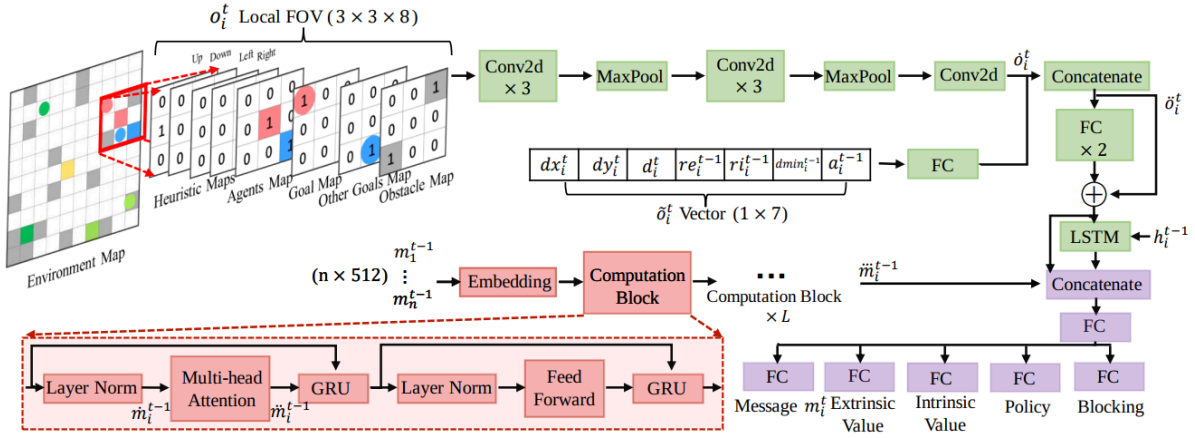

Рис. 5. Архитектура SCRIMP [1].

Она состоит из трех основных частей:

- *Модуль наблюдения*, или *observation encoder* (на картинке зеленый) является сверточной сетью, которая принимает на вход окружение агента в радиусе одной клетки, а также вектор из дополнительных характеристик агента.
- *Модуль коммуникации* (на картинке красный) отвечает за общение между агентами.
- *Выходной модуль*, или *output heads* (фиолетовый) совмещает результаты работы двух других модулей и принимает конечное решение о шаге агента.

Рассмотрим их подробнее. Следующие разделы подразумевают знакомство с основами глубокого обучения.

Модуль наблюдения

Его архитектура аналогична архитектуре сети в PRIMAL. Ключевое отличие заключается в дополнительных входах.

Агент напрямую наблюдать только зону 3×3 вокруг себя. Информация о расположении препятствий, наличии других агентов, и собственной цели, поступает напрямую в виде бинарных тензоров. Помимо этого, агент также получает на вход так называемые *эвристические карты* (heuristic maps) [7]. Их 4 — по одной для каждого направления сетки. Они показывают, правда ли, что если агент совершит шаг из клетки в соответствующем направлении, он приблизится к цели. Наконец, для всех остальных агентов в радиусе видимости, имеется бинарный тензор, описывающий направление этого агента к его цели.

Помимо этого, агент получает дополнительный вектор с информацией. В нем содержится направление до цели, евклидово расстояние, последние награды, полученные агентом, и последнее совершенное действие.

Входные карты проходят через сверточную сеть, затем смешиваются с дополнительным входным вектором, и результат записывается в LSTM ячейку памяти. Эта ячейка является основным состоянием агента. Выход LSTM передается в выходной модуль.

Модуль коммуникации

Каждый шаг, агент возвращает 512-мерный вектор — сообщение, которое получат другие агенты на следующем шаге. На первом шаге сообщение нулевое.

Эти сообщения (тензор $k \times 512$) централизовано (но независимо друг от друга) проходят через один блок трансформера-кодировщика после позиционного кодирования. Полученные эмбеддинги для каждого агента передаются выходному слою соответствующего агента.

Важно, что в данной архитектуре каждый агент общается с каждым. Важность сообщения определяет для себя сам агент, получивший сообщение, с помощью механизма внимания с 8-ю головами. Матрицы внимания можно интерпретировать как «интерес» агентов друг к другу.

Выходной модуль

Полученные выходы других модулей проходят через пару полносвязных слоев, чтобы получить политику (то есть следующее стохастическое действие агента), награды, и сообщение для следующего шага.

В SCRIMP используется несколько наград:
- Extrinsic reward является обычной наградой, которая дается за достижение цели и отбирается каждый шаг. Таким образом мы мотивируем агентов как можно быстрее двигаться к своим целям.
- Intrinsic reward дается за то, что агент изучает территорию карты, в которой его никогда не было (или давно не было).

- Blocking reward является штрафом за то, что из-за текущего расположения агента, какой-то другой агент не может найти путь короче.

Эти награды предсказываются моделью по отдельности.

Награда используемая при обучении является суммой этих наград.

Шаг вниз/вверх/влево/вправо	-0.3
Ожидание, не дойдя до цели	-0.3
Ожидание на конечной позиции	-0.0
Коллизия с другим агентом	-2.0
Блокирование другого агента	-1.0

Таблица 1. Структура наград SCRIMP

Взаимодействие со средой

SCRIMP сложным образом взаимодействует со средой.

Каждый шаг для каждого агента с помощью поиска в ширину находятся расстояния до цели из каждой клетки карты. Эти расстояния будут использованы для генерации эвристических карт.

После этого происходит непосредственно обращение к модели, из результата которой семплируется первое приближение следующего действия.

Для разрешения конфликтов используется сравнение предсказанных value-функций, то есть значений наград. Для конфликтующих агентов рассматриваются разные сценарии развития событий, и для каждого модель запускается заново, чтобы предсказать value на следующем шаге. «Победитель» в разрешении конфликта выбирается случайным образом. Логиты вероятностей пропорциональны изменению value между текущим шагом и следующими рассмотренными.

К сожалению, очевиден следующий парадокс: чтобы совершить следующий шаг в разных сценариях, необходимо разрешить конфликт, а это ровно то, чем мы сейчас занимаемся. В оригинальной реализации SCRIMP для этого используется специальная, сложная и запутанная логика. В нашей реализации конфликты предварительно разрешаются SAT солвером. Конечно, эта задача решается и за полиномиальное время (сведением к задаче о максимальном потоке), но в любом случае этот шаг не является «бутылочным горлышком», а структура задачи достаточно простая, чтобы ее быстро решил солвер.

Как только получены действия, не приводящие к конфликтам, они совершается.

Важной частью исполнения является *эпизодический буфер* (episodic buffer). Он хранит места, в которых недавно побывал агент. Новое место добавляется в буфер, если все остальные места достаточно далеки от него (хотя бы на $\tau = 3$). С помощью этого буфера вычисляется intrinsic reward, по формуле $\text{reward} = \varphi(\beta - \delta)$, где $\varphi = 0.2$, $\beta = 1$, а δ является индикатором того, что все позиции в буфере удалены от текущей хотя бы на $\tau = 3$.

Чтобы вычислить blocking reward, для каждого агента в радиусе видимости запускается A* для поиска пути к цели, и сравнивается расстояние в случае, когда агент есть, и когда его нет. Если второе меньше, агент получает штраф.

Обучение

Мы не обучали модели. Мы реализовали всю логику, связанную непосредственно с поиском: модели, окружения, логику взаимодействия, а также инструменты для работы с этим.

SCRIMP обучается с помощью Clip-PPO и имитации. В качестве эксперта выступает ODrM* [8]. Карты генерируются случайным образом, препятствия распределены равномерно. В каждой клетке вероятность препятствия задается треугольным распределением. Расположение начальных и конечных позиций случайно выбирается из наибольшей связной области на карте.

У нас не было вычислительных ресурсов и свободного времени для реализации качественного обучения, поэтому мы использовали готовые веса. Мы реализовали скрипт для автоматической конвертации весов для оригинального SCRIMP в нашу реализацию, проверяя корректность, и использовали их.

DCC

DCC (Decision Causal Communication) предлагает решение проблемы избыточной коммуникации между агентами с помощью обучаемой выборочной коммуникации

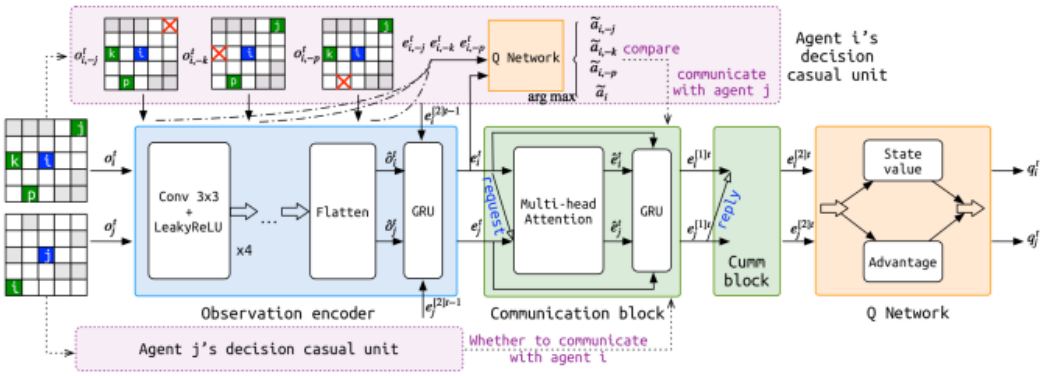

Рис. 6. Архитектура DCC [2].

Модель состоит из нескольких блоков:

- Observation encoder - На вход передается 6 «слоев» бинарной информации: карта препятствий , карта других агентов и 4 карты эвристик (1-чки в этих картах говорят, что ход (N/S/E/W) из этой клетки уменьшит агенту кратчайшее расстояние до цели, найденное алгоритмом A* на карте без других агентов), все сужено до FOV. Сам энкодер состоит из нескольких последовательных сверток и GRU в конце
- Decision causal unit - Пробуем выкинуть каждого из соседних агентов из FOV и используем Q-сеть для предсказания следующего хода в каждом из таких случаев и не выкидывая никого. Агенты с которыми нужна коммуникация - в точности те, выкидывание которых меняет предсказанный следующий ход.
- Communication block - Графовая свертка с трансформером в качестве ядра свертки и GRU в конце. Таких блоков 2 - один для исходящих сообщений и второй для ответных
- Q-сеть - Для предсказания следующего хода. Награды (reward) выглядят следующим образом:

Шаг вниз/вверх/влево/вправо	−0.075
Ожидание, не дойдя до цели	−0.075
Ожидание на конечной позиции	0
Коллизия с другим агентом	−0.5
Финиш	3.0

Таблица 2. Структура наград DCC

Обучается модель как распределенный Deep Q-learning с расписанием (curriculum learning), процесс генерации карт очень похож на описанный ниже. Идея расписания заключается в следующем: хочется, чтобы модель научилась решать сложные задачи с большим количеством агентов, но если начать сразу учиться на них, то обучение будет очень медленным и нестабильным, поэтому вместо этого обучение начинается на простых примерах (1 агент и карта 10x10) и сложность прогрессивно увеличивается (в терминах размера карт и количества агентов) как только вероятность успешного завершения карты текущей сложности превышает определенный порог (0.9 у авторов оригинальной статьи).

Отметим, что следуя методологии авторов, DCC в нашей реализации не совершает вообще никаких конфликтов из рисунка 3. Корректность решения сохраняется. Также, в нашей реализации конфликты разрешаются прямым приоритезированием левого-верхнего агента.

Методология

Мы исследовали поведение моделей и собирали данные на случайно сгенерированных картах.

Каждая карта задается 3-я числами: линейным размером (для простоты все карты мы брали квадратными, как в оригинальных статьях), количеством агентов и плотностью препятствий. Для начала генерировались препятствия из распределения Бернулли (каждая клетка карты с вероятностью равной плотности препятствий выбирается препятствием), затем оставшееся свободное место карты разбивается на компоненты связности (чтобы не генерировать карт, где агент не может добраться до цели исключительно из-за препятствий), для каждого агента выбирается «своя» компонента связности, но в каждой компоненте связности не может быть больше агентов, чем половина ее площади, и затем внутри этой компоненты выбираются непересекающиеся начальные и конечные позиции агентов. Если после разбиения на компоненты связности не хватало места для всех агентов, то карта перегенерировалась. Такой алгоритм генерации не гарантирует разрешимость задачи MAPF, но гарантирует, что плотность агентов в любой компоненте связности будет не больше 0.5.

С его помощью был создан датасет из 30 карт 36 разных видов: с плотностями 0.2, 0.3 и 0.4, размерами 32, 64 и 96, и количеством агентов 16, 32, 64 и 96. DCC был протестирован на всех таких картах. SCRIMP только на картах 32x32 и 64x64. На каждой карте агент запускался единожды, в детерминированной, воспроизводимой[1] среде. Если решение не было найдено за 512 шагов, поиск прекращался, и запуск считается неудачным.

Карты были выбраны таким образом, потому что их легко массово генерировать и собирать агрегированные результаты, они повторяют задачи, на которых модели обучались. Карты, созданные вручную также рассматривались, но только для ручного анализа и визуализации. В агрегированных результатах их нет.

На основе статистик, приведенных в соответствующих статьях, ожидается, что SCRIMP покажет себя немного лучше DCC.

[1] без учета недетерминизма, вызванного особенностями железа и реализацией генераторов случайных чисел.

Анализ

В таблице 3 представлены вероятности успеха прохождения DCC различных карт. Можно обратить внимание, что вероятность успеха по большому счету определяется плотностью агентов, а не их количеством. Это логично: каждый агент принимает решение только на основе того, что он видит вокруг себя, и механизм общения недостаточно выразителен, чтобы решать сложные конфликты на месте. На больших разреженных картах зачастую агентам не требуется общаться совсем, и задача превращается в обычный поиск пути, а с ней модель справляется прекрасно за счет эвристических карт.

	Количество агентов											
Размер карты	**16**			**32**			**64**			**96**		
32x32	100%	100%	93%	100%	97%	87%	93%	67%	17%	57%	13%	0%
64x64	97%	100%	97%	97%	100%	87%	100%	97%	80%	93%	90%	40%
96x96	97%	100%	100%	97%	100%	100%	80%	93%	100%	83%	97%	80%
	0.2	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**
	Плотность препятствий											

Таблица 3. Вероятности успеха DCC на разных картах.

SCRIMP показывает результаты немного лучше на маленьких картах, как показано в таблице 4.

	Количество агентов											
Размер карты	**16**			**32**			**64**			**96**		
32x32	100%	100%	100%	100%	93%	87%	93%	73%	40%	60%	27%	3%
64x64	100%	100%	100%	100%	100%	93%	97%	93%	77%	93%	80%	???
	0.2	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**
	Плотность препятствий											

Таблица 4. Вероятности успеха SCRIMP

Посмотрим на makespan среди **решенных** задач: таблицы 5 и 6. Здесь оказывается, что SCRIMP снова в среднем побеждает DCC на маленьких картах, но немного проигрывает на больших.

	Количество агентов											
Размер карты	**16**			**32**			**64**			**96**		
32x32	45.6	61.8	57.4	64.8	79.4	114.7	111.1	166.1	208.0	182.8	278.0	???
64x64	91.3	96.2	113.9	98.1	116.4	141.5	111.3	131.6	206.4	115.9	166.5	295.1
96x96	132.9	139.2	205.1	161.3	166.9	232.4	163.3	160.2	258.1	177.8	187.6	296.2
	0.2	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**
	Плотность препятствий											

Таблица 5. Средний makespan (время последнего шага) среди решенных задач. DCC. Зеленым цветом обозначены сценарии, где результат оказался лучше, чем у SCRIMP.

	Количество агентов											
Размер карты	**16**			**32**			**64**			**96**		
32x32	44.0	62.0	64.0	62.6	92.0	112.2	82.6	169.7	186.4	176.8	222.2	297.0
64x64	89.7	100.7	114.5	100.7	123.3	147.8	124.9	153.5	184.4	147.1	187.3	???
	0.2	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**
	Плотность препятствий											

Таблица 6. Средний makespan (время последнего шага) среди решенных задач. SCRIMP

Тут я хочу немного остановиться, и описать, почему мы вообще это наблюдаем. В теории, SCRIMP является более выразительной моделью. Его механизм коммуникации позволяет любому агенту общаться с любым другим агентом. Это компенсирует маленький FOV (радиус видимости) — в любом случае гораздо сложнее уворачиваться от других агентов, чем от статических препятствий, особенно на картах нашей природы, когда нету нетривиальных тупиков, и основная сложность задачи — правильно оркестрировать агентов. Поэтому разумно ожидать, что SCRIMP будет немного лучше себя показывать на тех картах, где коммуникация дает значительный бонус из-за плотности агентов, а на разреженных картах, где важнее прокладывать маршрут, лучше справляется DCC.

Более того, SCRIMP получает награду за дополнительное исследование, а DCC нет. Поэтому если исследование является ключевым моментов в решении, SCRIMP снова должен побеждать.

Разумеется, такая же ситуация должна быть со средним временем за которое агент достигает цели, да?

	Количество агентов											
Размер карты	**16**			**32**			**64**			**96**		
32x32	21.3	30.6	23.0	26.4	35.2	36.6	34.7	65.5	73.3	48.3	81.2	???
64x64	44.5	52.3	42.0	44.9	53.9	50.2	47.5	60.4	67.7	50.7	65.4	110.6
96x96	68.7	74.0	77.1	67.1	77.0	87.8	70.3	78.7	84.9	70.1	84.0	87.9
	0.2	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**
	Плотность препятствий											

Таблица 7. Среднее время для достижения агентом цели, DCC

	Количество агентов											
Размер карты	**16**			**32**			**64**			**96**		
32x32	23.5	34.1	28.1	30.4	42.8	43.0	37.1	67.5	73.2	64.2	87.6	117.1
64x64	50.4	60.3	56.3	51.1	63.2	66.2	55.0	70.9	72.5	62.2	82.6	???
	0.2	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**
	Плотность препятствий											

Таблица 8. Среднее время для достижения агентом цели, SCRIMP

Оказывается, что нет. Скорее всего это вызвано тем, что у SCRIMP больше механизмов для того, чтобы предотвращать ситуации, когда агент на конечной позиции блокирует другого, поэтому агенты чаще сходят со своих целей.

В подтверждение этой гипотезы, можно сравнить первый момент времени, когда агент доходит до цели.

Размер карты	Количество агентов: 16			32			64			96		
32x32	20.9	28.7	20.2	24.5	30.3	28.7	29.0	52.5	51.0	36.7	56.8	???
64x64	43.8	51.4	40.1	43.8	51.9	45.4	45.5	55.9	59.3	47.8	58.9	91.0
96x96	67.9	72.7	74.7	66.1	74.6	82.0	68.6	74.6	79.1	67.5	79.8	80.2
	0.2	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**
						Плотность препятствий						

Таблица 9. Первый момент времени, когда агент находится на конечной позиции, DCC

Размер карты	Количество агентов: 16			32			64			96		
32x32	20.5	27.6	18.4	23.2	26.6	23.0	25.6	35.7	27.2	27.4	35.0	27.0
64x64	43.4	51.5	39.2	43.5	51.0	45.9	45.3	53.5	48.0	46.9	55.5	???
	0.2	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**
						Плотность препятствий						

Таблица 10. Первый момент времени, когда агент находится на конечной позиции, SCRIMP

Действительно, SCRIMP на самом деле быстрее, но любит менять свое мнение.

Посмотрим на распределения этих характеристик.

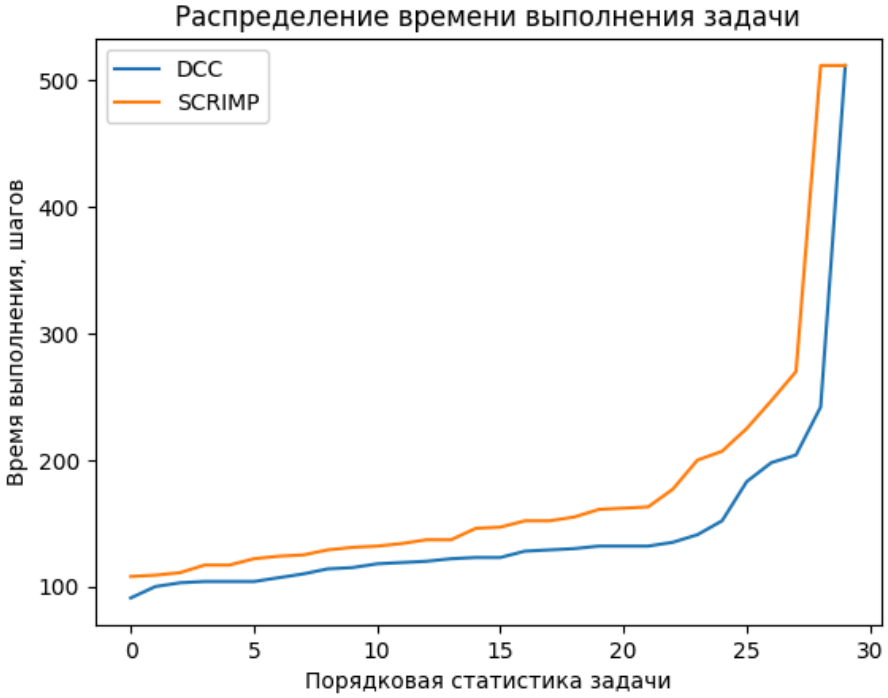

Рис. 7. Распределение makespan-ов алгоритмов на картах 64x64, с 64 агентами, плотность препятствий 0.3. Значения на графике соответствуют порядковым статистикам (или квантилям). Площадь под графиком — среднему.

Рисунок 7 показывает распределение makespan для решения нескольких карт. Мы уже знаем, что SCRIMP проигрывает на картах такого размера, и теперь можем в среднем прикинуть, на сколько. Важно, что такой график не показывает время работы на отдельных тестах, а сорти-

рует и перемешивает их, поэтому это не означает, что SCRIMP всегда хуже, но означает, что он хуже в среднем.

Также заметно, что с большинством задач модели справляются хорошо, за исключением нескольких сложных выбросов, причем согласно рисунку 8, разных. Это может намекать на положительный эффект ансемблирования применительно к задаче.

Рис. 8. Распределение makespan-ов алгоритмов на картах 64x64, с 64 агентами, плотность препятствий 0.3. Значения соответствуют задачам.

Наконец, посмотрим на особые характеристики DCC, а именно, количество коммуникаций.

	Количество агентов											
Размер карты	**16**			**32**			**64**			**96**		
32x32	37.8	78.0	90.1	197.1	393.9	505.3	1133.9	3239.4	3441.0	3454.4	6859.2	???
64x64	20.2	30.2	43.0	82.9	146.2	221.9	347.4	731.9	1858.5	909.4	1717.2	6642.6
96x96	16.1	15.9	53.0	59.4	94.1	256.6	248.9	374.0	1067.7	509.7	1035.4	3028.1
	0.2	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**
	Плотность препятствий											

Таблица 11. Среднее количество коммуникаций в DCC среди успешно пройденных карт.

	Количество агентов											
Размер карты	**16**			**32**			**64**			**96**		
32x32	0.0184	0.0185	0.0193	0.0385	0.0659	0.0482	0.0941	0.1995	0.0403	0.1117	0.0670	0.1080
64x64	0.0026	0.0025	0.0037	0.0089	0.0103	0.0137	0.0277	0.0512	0.0275	0.0697	0.0334	0.1379
96x96	0.0009	0.0007	0.0015	0.0007	0.0038	0.0022	0.0084	0.0135	0.0221	0.0177	0.0320	0.0257
	0.2	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**	**0.2**	**0.3**	**0.4**
	Плотность препятствий											

Таблица 12. Среднее количество коммуникаций на одного агента на один шаг в DCC среди всех карт.

Заметно, что на больших картах коммуникации случаются гораздо реже, чем на маленьких. Это ожидаемо: на больших картах агенты справляются дойти до целей самостоятельно.

Последнее, что нам хочется показать, это attention матрицы в SCRIMP. Их можно интерпретировать, как интерес агентов друг к другу. Хотя в начале матрицы выглядят почли случайно, видно, как ближе к концу решения, агенты активно обращают внимание на агентов, которые пока не дошли до цели.

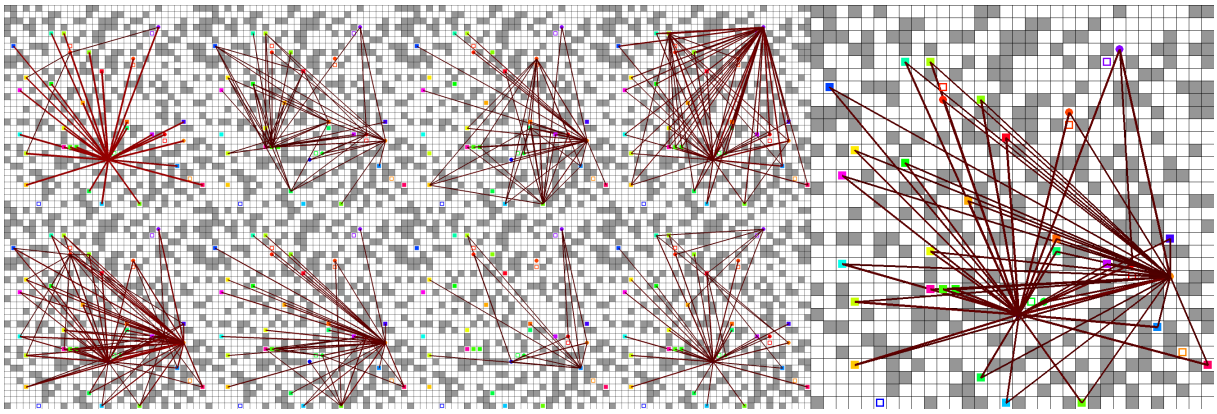

Рис. 9. Attention агентов в SCRIMP по отношению друг к другу ближе к конце решения задачи. Слева: 8 матриц attention-а каждой из головы трансформера. Справа: усредненное внимание по всем головам. Отрезки соединяют агентов, у которых между собой большой attention, и чем отрезок толще и краснее, тем больше значение. Видно, как агенты активно обращают внимание на агентов, которые пока не дошли до цели

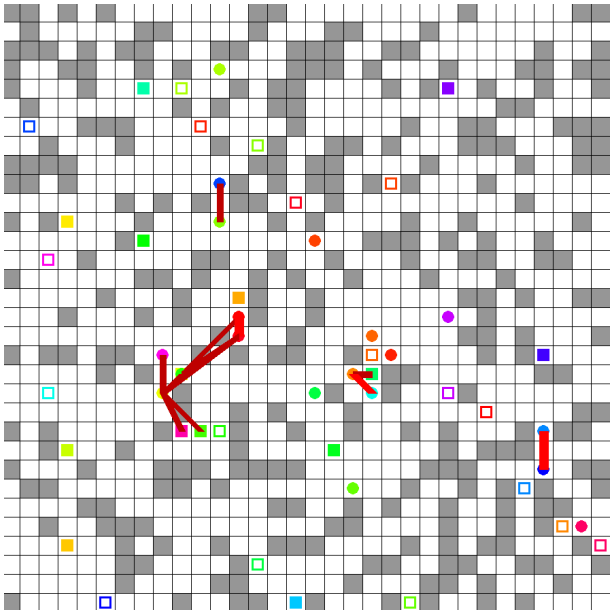

Рис. 10. Коммуникация между агентами в процессе решения в DCC. Показана для сравнения с SCRIMP.

Заключение

Мы реализовали и сравнили модели SCRIMP и DCC. Оказалось, что SCRIMP показывает себя лучше на небольших картах, и проигрывает на картах побольше. Были озвучены предположительные причины, почему. Были сравнены метрики makespan и flowtime. На глаз, результаты соответствуют результатам из оригинальных статей.

Во время работы над проектом, возникло несколько идей по возможному улучшению результатов. Вот некоторые из них:

- Ансемблирование: обучение нескольких моделей для разных ситуаций, и их параллельный запуск для получения более качественных результатов.
- Предиктивные награды: дообучить модель предсказывать ее положения в будущем, чтобы мотивировать модель «думать наперед». Также может быть использовано для разрешения конфликтов.
- Beam Search: рассматривать одновременно несколько траекторий, и по достижению цели выбирать оптимальную.
- Архитектурные улучшения: использовать более современные методы в архитектурах самих моделей.

Библиография

[1] Y. Wang, B. Xiang, S. Huang, и G. Sartoretti, «SCRIMP: Scalable Communication for Reinforcement- and Imitation-Learning-Based Multi-Agent Pathfinding». Просмотрено: 28 январь 2025 г. [Онлайн]. Доступно на: http://arxiv.org/abs/2303.00605

[2] Z. Ma, Y. Luo, и J. Pan, «Learning Selective Communication for Multi-Agent Path Finding». Просмотрено: 28 январь 2025 г. [Онлайн]. Доступно на: http://arxiv.org/abs/2109.05413

[3] R. Stern *и др.*, «Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks». Просмотрено: 28 январь 2025 г. [Онлайн]. Доступно на: http://arxiv.org/abs/1906.08291

[4] R. Stern, «Multi-Agent Path Finding – An Overview», *Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, сс. 96–115, 11 март 2022 г. Просмотрено: 28 январь 2025 г. [Онлайн]. Доступно на: https://doi.org/10.1007/978-3-030-33274-7_6

[5] G. Sartoretti *и др.*, «PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning». Просмотрено: 28 январь 2025 г. [Онлайн]. Доступно на: http://arxiv.org/abs/1809.03531

[6] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, и P. Abbeel, «Overcoming Exploration in Reinforcement Learning with Demonstrations». Просмотрено: 28 январь 2025 г. [Онлайн]. Доступно на: http://arxiv.org/abs/1709.10089

[7] Z. Ma, Y. Luo, и H. Ma, «Distributed Heuristic Multi-Agent Path Finding with Communication». Просмотрено: 28 январь 2025 г. [Онлайн]. Доступно на: http://arxiv.org/abs/2106.11365

[8] C. Ferner, G. Wagner, и H. Choset, «ODrM* optimal multirobot path planning in low dimensional search spaces», в *2013 IEEE International Conference on Robotics and Automation*, май 2013, сс. 3854–3859. doi: 10.1109/ICRA.2013.6631119.