Московский Физико Технический Институт
Центр Когнитивного Моделирования

Питанов Елисей Сергеевич

«MAPF»

Отчет по курсу "Эвристические методы планирования"

Москва, 2022

Содержание

1. Введение

В наше время транспорт стал неотъемлемой частью повседневной жизни. Он помогает нам добираться на работу, с помощью него нам привозят необходимые вещи, он используется в коммерции для логистики грузов как на малые, так и большие расстояния. Тот же транспорт при правильном выборе и использовании может даже доставлять удовольствие от управления им. Однако, в этой работе остановимся на в большей части коммерческом транспорте, или же каком то виде личного. Все больше и больше транспорта оснащается какими то видами автопилота, и пусть сейчас его возможности весьма ограничены, близок тот день, когда большая часть транспорта окажется под управлением не человека, но сложной вычислительной системы. И тогда остро встает вопрос - как осуществить и создать максимально безопасное движение огромного количества агентов движения на каком то ограниченном пространстве? Здесь нам приходит на помощь относительно молодой раздел компьютерных наук - планирование. А точнее - планирование с помощью современных подходов, таких как нейронные сети или обучение с подкреплением. Задача усложняется тем, что планирование необходимо осуществить не только для одного объекта - участника движения, но и для всех в том числе. Таким образом формируется задача **многоагентного планирования** - необходимо за максимально малое количество условных шагов добиться того, чтобы каждый агент достиг своей цели. Примером такого случая может послужить движение автомобилей на одном из самых сложных перекрестков мира - перекресток на площади Мексель, Эфиопия(рис. 1.1). Необходимо не допустить аварий и при этом каждый автомобиль должен достигнуть нужного выезда с перекрестка. Опытный человек справляется с этой задачей достаточно успешно, так как аварии на данном участке достаточно редки при высокой интенсивности движения, однако, что же будет, если человека программно заменить?

2. Постановка задачи

Задача MAPF(Multi-agent pathfinding) формулируется следующим образом:

- Граф G(V,E) - состояние среды
- Набор агентов $A_1(s_1, g_1)$ .. $A_k(s_k, g_k)$
- Набор действий Actions
- План для агента A_i последовательность действий π_i такая, что при последовательном выполнении π_i агентом A_i он достигнет своей цели g_i, избежав все препятствия, как статические, так и динамические. Совокупность планов для всех агентов - **общий план**.
- А так же вводится *cost* - стоимость прохождения плана.
- Итого задача - найти:

$$plan = \arg\min_i(\sum_j cost(plan_{j_i})) \tag{2.1}$$

Рис. 1.1: Перекресток на площади Мексель

где $plan_{j_i}$ - ій план в среде для jго агента

Используемая метрика: CSR = 1, если все агенты в среде дошли до цели, 0 иначе

При этом в данном проекте будет исследован вариант с частично наблюдаемой средой, при этом, частично наблюдаемый марковский случайный процесс (POMDP) для одного агента записывается в виде $\langle S, A, O, T, p, r, \mathcal{I}, p_o, \gamma \rangle$, где:

- S – множество состояний среды,
- A – множество доступных действий,
- O – множество наблюдений,
- $T : S \times A \to S$ – функция переходов,
- $p(s'|s, a)$ – вероятность перехода в состояние s' из состояния s при действии a,
- $r : S \times A \to \mathbb{R}$ – функция вознаграждения,
- $\mathcal{I} : S \to O$ – функция наблюдения,
- $p_o(o|s', a)$ – вероятность получить наблюдение o, если был совершен переход в состояние s' при предпринятом действии a,
- $\gamma \in [0, 1]$ – фактор дисконтирования.

Многоагентный POMDP может быть представлен как $\langle S, U, P, r, Z, O, n, \gamma \rangle$

```
1  g(s_start) = 0; OPEN = ∅;
2  insert s_start into OPEN with f(s_start) = h(s_start);
3  while(s_goal is not expanded)
4    remove s with the smallest f-value from OPEN;
5    successors = getSuccessors(s);
6    for each s' in successors
7      if s' was not visited before then
8        f(s') = g(s') = ∞;
9      if g(s') > g(s) + c(s, s')
10       g(s') = g(s) + c(s, s');
11       updateTime(s');
12       f(s') = g(s') + h(s');
13       insert s' into OPEN with f(s');
```

Рис. 3.1: Safe A*

- $s \in S$ описывает текущее состояние среды. В каждый момент времени каждый агент $a \in A\{1..n\}$ выбирает действие $u^a \in U$, формируя объединенное действие $\mathbf{u} \in \mathbf{U}$. Это вызывает изменения в среде в соответствии с функцией перехода $P(s'|s,u) : S \times U \times S \to [0,1]$.
- Все агенты используют одну и ту же функцию вознаграждения $r(s,u) : S \times U \to R$ с γ - дисконтирующим фактором.
- Частичная наблюдаемость обеспечивается тем, что у каждого агента есть собственные наблюдения $z \in Z$, соответственные функции наблюдений $O(s,a) : S \times A \to Z$.
- Каждый агент хранит историю действий $\tau^a \in T = (Z \times U)^*$, на основании которых он выводит стохастическую стратегию $\pi^a(u^a|\tau^a) : T \times U \to [0,1]$.

3. Известные алгоритмы

Алгоритмы в данной задаче делятся на 2 типа:

- Централизованные - каждый агент планируется внутри общей схемы
- Децентрализованные - каждый агент воспринимает других как динамические препятствия

Среди централизованных алгоритмов наиболее простым является Safe - A*[2], который, оперируя понятием времени и безопасного интеравала, разрешает коллизии(3.1, 3.2)

Децентрализованные алгоритмы же обычно более сложные, и нередко используют в своей основе нейросеть. Например, метод[1] использует сети разных типов для разрешения конфликтов и коммуникации между агентами3.3.

4. Предлагаемый алгоритм

Для RL - задач часто используется алгоритм PPO[4], который хорошо работает для такого рода условий, однако плохо разрешает сложные конфликты. Предлагается протестировать комбинацию PPO и Coop A* алгоритмов. Общая идея:

- Используется APPO

```
1  getSuccessors(s)
2    successors = ∅;
3    for each m in M(s)
4      cfg =configuration of m applied to s
5      m_time =time to execute m
6      start_t = time(s) + m_time
7      end_t = endTime(interval(s)) + m_time
8      for each safe interval i in cfg
9        if startTime(i) > end_t or endTime(i) < start_t
10         continue
11       t =earliest arrival time at cfg during interval i with no collisions
12       if t does not exist
13         continue
14       s' = state of configuration cfg with interval i and time t
15       insert s' into successors
16   return successors;
```

Рис. 3.2: Safe A*: get successors

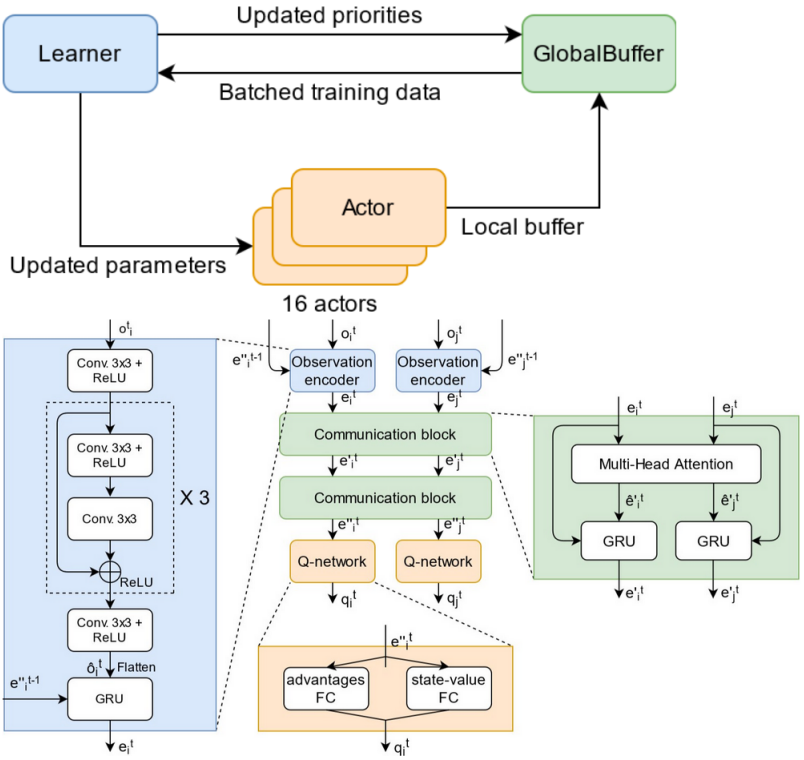

Рис. 3.3: DHL

- При появлении в локальной окрестности агента других агентов применяется Coop A*
- Тестирование в среде Pogema
- Сравнение с "чистым" APPO по числу итераций в среде, FPS и CSR

5. Программная реализация

Данный алгоритм был реализован с помощью библиотек pogema, gym и sample_factory. Из последней была взята реализация APPO. Код выложен в репозитории на github и программно состоит из 4 файлов.

1) appo.py - определение класса APPO
2) astar.py - реализация Coop A*
3) asappo.py - определение гибридного алгоритма
4) main.py - модуль запуска эксперментов

6. Результаты экспериментов

В качестве бейзлайн - алгоритма APPO были взяты веса, полученные в результате обучения в Curriculum learning[3]. Было зафиксировано несколько конфигураций среды, и проведены серии экспериментов с различным random seed. Измерены средние значения CSR, ISR, FPS, makespan. Размер наблюдаемой окрестности был равен 5 во всех случаях. A* применялся, если расстояние между агентами было равно 3.

Алгоритм	Кол-во агентов	Размер среды	Пл-ть препятствий	Число шагов	CSR	ISR	FPS	makespan
APPO	24	12	0.3	32	0.08	0.7	64	31.62
ASAPPO	24	12	0.3	32	0.14	0.72	61.26	31.16
APPO	32	16	0.3	64	0.38	0.84125	47.47	60.02
ASAPPO	32	16	0.3	64	0.42	0.87125	45.51	59.7
APPO	64	32	0.3	128	0.2	0.89	18.29	123.02
ASAPPO	64	32	0.3	128	0.22	0.91	17.66	124.12

7. Выводы

Таким образом, предложенный алгоритм показал свою эффективность для разрешения конфликтов. Можно заметить, что на всех тестируемых конфигурациях был получен прирост по метрикам при небольшом падении FPS. Наиболее явно этот эффект прослеживается на больших картах и с большим числом агентов. При этом плотность препятствий, примененная в экспериментах, способствует нахождению целей на достаточно большом удалении от точки старта агента, и при этом

образованию "коридорных"конфликтов, с которыми полученных гибридный алгоритм справляется лучше бейзлайна. Из минусов хотелось бы отметить, что такой алгоритм предполагает наличие связи между агентами в пределах конфликтной окрестноси.

Список литературы

[1] Ziyuan Ma, Yudong Luo, and Hang Ma. Distributed heuristic multi-agent path finding with communication. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8699–8705. IEEE, 2021.

[2] Mike Phillips and Maxim Likhachev. Sipp: Safe interval path planning for dynamic environments. In *2011 IEEE International Conference on Robotics and Automation*, pages 5628–5635. IEEE, 2011.

[3] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.