MAPF

Питанов Е.С.
pitanov.es@phystech.edu

Центр когнитивного моделирования МФТИ

Мотивация

Все больше и больше транспортных средств становятся беспилотными, а значит нужно будет управлять их большим количеством. И здесь не обойтись без планирования - возможности предложить максимально эффективные действия без исполнения их в реальной среде.

Рис.: Перекресток на площади Мексель, Эфиопия

Обобщенная постановка

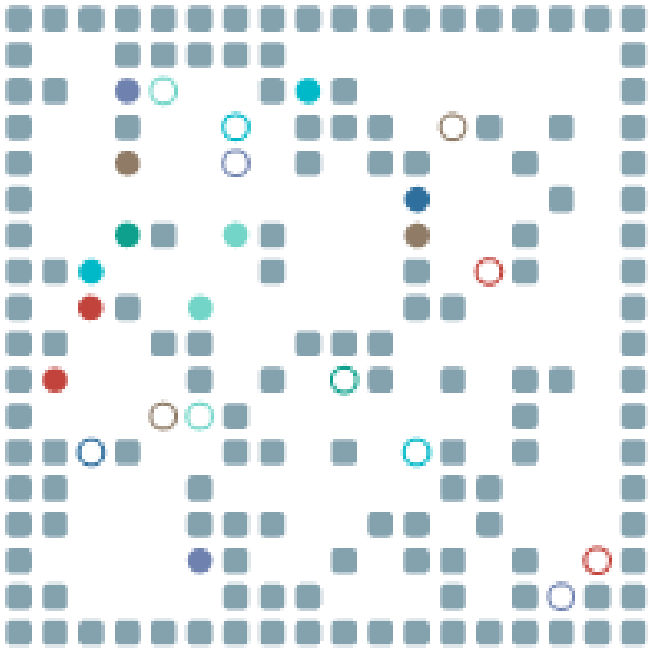

Рис.: Пример задачи MAPF

Обобщенная постановка

Задача MAPF:

- 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. Сосокупность планов для всех агентов - **общий план**.
- А так же вводится *cost* - стоимость прохождения плана.
- Итого задача - найти:

$$plan = \arg\min_{i}(\sum_{j} cost(plan_{j_i})) \quad (1)$$

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

Обобщенная постановка

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

Обобщенная постановка

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.

Обобщенная постановка

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

- $s \in S$ 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 $\pi^a(u^a|\tau^a) : T \times U \to [0,1]$.

Centralized

```
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.: Safe A*

```
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;
```

Decentralized

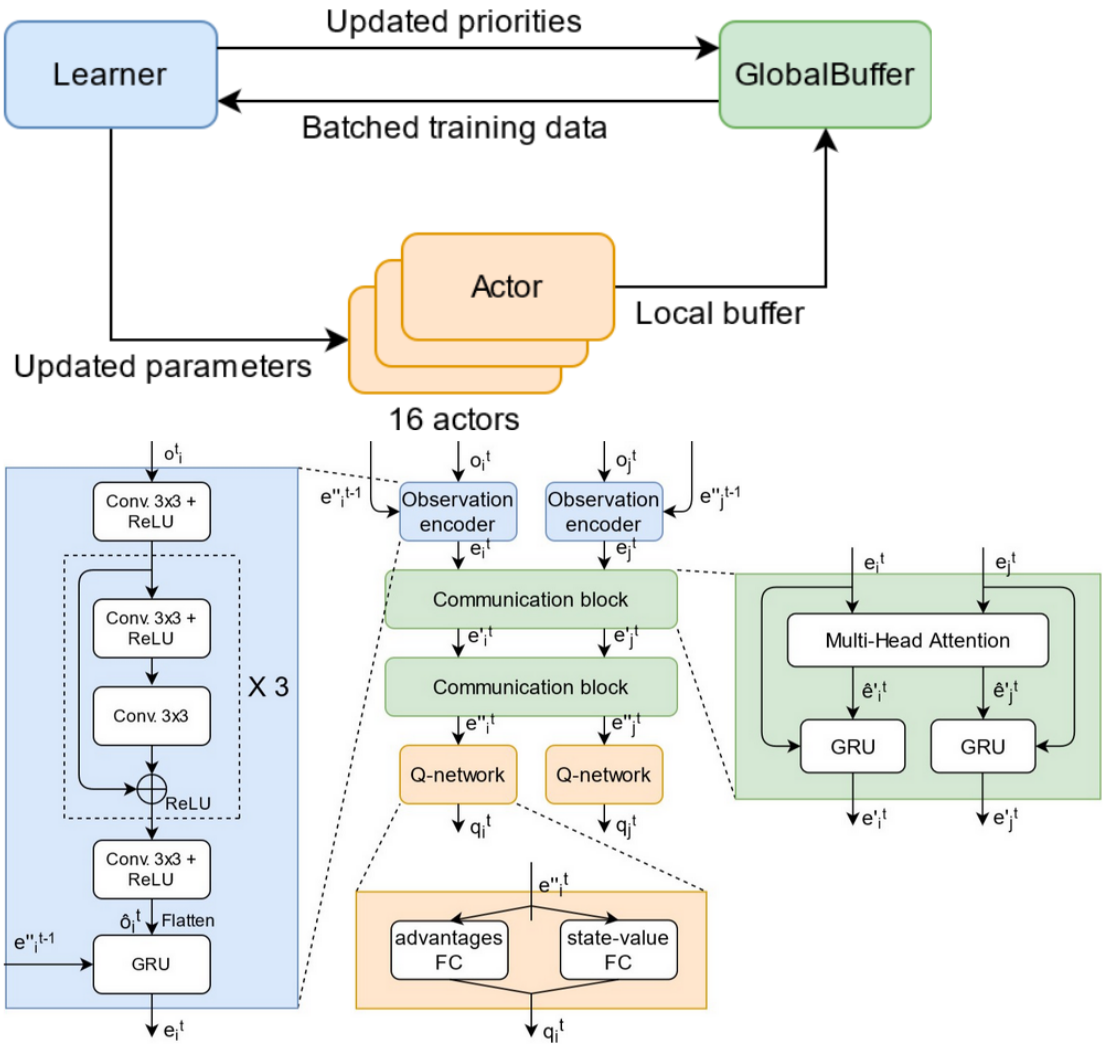

Рис.: DHC

Предлагаемый метод

Идея алгоритма:

- Используется APPO
- 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

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

Сводная таблица

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

Выводы

- Алгоритм показал эффективность в разрешении "коридорных" конфликтов по сравнению с бейзлайном
- Алгоритм предполагает наличие связи между агентами в конфликтной окрестности

Спасибо за внимание!

cogmodel.mipt.ru
rairi.ru
raai.org

pitanov.es@phystech.edu