

第十三章 多智能体系统

之前章节的设定都是单智能体系统 (Single-Agent System, 缩写 SAS)。本章和后面三章介绍多智能体系统 (Multi-Agent System, 缩写 MAS) 和多智能体强化学习 (Multi-Agent Reinforcement Learning, 缩写 MARL)。本章讲解多智能体系统的基本概念，帮助大家理解 MAS 与 SAS 的区别。第 13.1 节讲解 MAS 的四种常见设定。第 13.2 节定义多 MAS 的专业术语，将之前所学的观测、动作、奖励、策略、价值等概念推广到 MAS。第 13.3 节介绍几种常用的实验环境，用于对比 MARL 方法的优劣。

13.1 多智能体系统的设定

多智能体系统与单智能体系统的区别： 多智能体系统 (Multi-Agent System, 缩写 MAS) 中包含 m 个智能体，智能体共享环境，智能体之间会相互影响。智能体之间是如何相互影响的呢？一个智能体的动作会改变环境状态，从而影响其余所有智能体。举个例子，股市中的每个自动交易程序就可以看做一个智能体。尽管智能体（自动交易程序）之间不会相互交流，它们依然会相互影响：一个交易程序的决策会影响股价，从而对其他自动交易程序有利或有害。

注意，MAS 与上一章的并行强化学习是不同的概念。上一章用 m 个节点并行计算，每个节点有独立的环境，每个环境中有一个智能体。虽然 m 个节点上一共有 m 个智能体，但是智能体之间完全独立，不会相互影响。而本章 MAS 只有一个环境，环境中有 m 个相互影响的智能体。并行强化学习的设定是 m 个单智能体系统 (Single-Agent System, 缩写 SAS) 的并集，可以视作 MAS 的一种特例。举个例子，环境中有 m 个机器人，这属于 MAS 的设定。假如把每个机器人隔绝在一个密闭的房间中，机器人之间不会通信，那么 MAS 就变成了多个 SAS 的并集。

多智能体强化学习 (Multi-Agent Reinforcement Learning, 缩写 MARL) 是指让多个智能体处于相同的环境中，每个智能体独立与环境交互，利用环境反馈的奖励改进自己的策略，以获得更高的回报（即累计奖励）。在多智能体系统中，一个智能体的策略不能简单依赖于自身的观测、动作、奖励，还需要考虑到其他智能体的观测和动作。因此，MARL 比单智能体强化学习 (Single-Agent Reinforcement Learning, 缩写 SARL) 更困难。

多智能体系统有四种常见设定： 合作关系 (Fully Cooperative)、竞争关系 (Fully Competitive)、合作竞争的混合 (Mixed Cooperative & Competitive)、利己主义 (Self-Interested)。图 13.1 举例说明了四种常见设定。接下来具体讲解这些设定。

第一种设定是完全合作关系：智能体的利益一致，获得的奖励相同，有共同的目标。比如图 13.1 中，多个工业机器人协同装配汽车。他们的目标是相同的，都希望把汽车装好。假设一共有 m 个智能体，它们在 t 时刻获得的奖励分别是 $R_t^1, R_t^2, \dots, R_t^m$ 。（用上标表示智能体，用下标表示时刻。）在完全合作关系中，它们的奖励是相同的：

$$R_t^1 = R_t^2 = \dots = R_t^m, \quad \forall t.$$

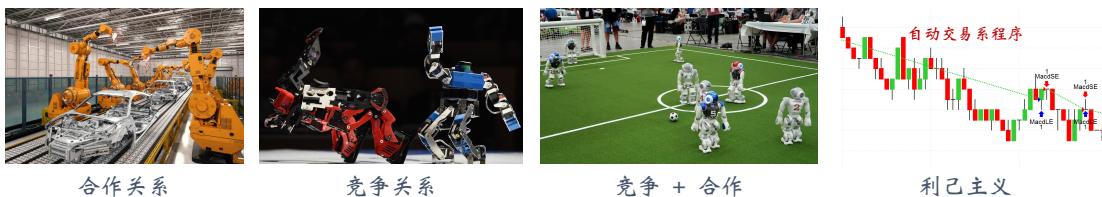


图 13.1: 多智能体强化学习的四种常见设定。四张图片来源于网络。

第二种设定是**完全竞争关系**: 一方的收益是另一方的损失。比如图 13.1 中的两个格斗机器人，它们的利益是冲突的，一方的胜利就是另一方的失败。在完全竞争的设定下，双方的奖励是负相关的：对于所有的 t ，有 $R_t^1 \propto -R_t^2$ 。如果是零和博弈，双方的获得的奖励总和等于 0： $R_t^1 = -R_t^2$ 。

第三种设定是**合作竞争的混合**。智能体分成多个群组；组内的智能体是合作关系，它们的奖励相同；组间是竞争关系，两组的奖励是负相关的。比如图 13.1 中的足球机器人：两组是竞争关系，一方的进球是另一方的损失；而组内是合作关系，队友的利益是一致的。

第四种设定是**利己主义**。系统内有多个智能体；一个智能体的动作会改变环境状态，从而让别的智能体受益或者受损。利己主义的意思是智能体只想最大化自身的累计奖励，而不在乎他人收益或者受损。比如图 13.1 中的股票自动交易程序可以看做是一个智能体；环境（股市）中有多个智能体。这些智能体的目标都是最大化自身的收益，因此可以看做利己主义。智能体之间会相互影响：一个智能体的决策会影响股价，从而影响其他自动交易程序的收益。智能体之间有潜在而又未知的竞争与合作关系：一个智能体的决策可能会帮助其他智能体获利，也可能导致其他智能体受损。设计自动交易程序的时候，不应当把它看做孤立的系统，而应当考虑到其他自动交易程序的行为。

不同设定下学出的策略会有所不同。在**合作**的设定下，每个智能体的决策要考虑到队友的策略，要与队友做到尽量好的配合，而不是个人英雄主义；这个道理在足球、竞技电游中是显然的。在**竞争**的设定下，智能体要考虑到对手的策略，相应调整自身策略；比如在象棋游戏中，如果你很熟悉对手的套路，并相应调整自己的策略，那么你的胜算会更大。在**利己主义**的设定下，一个智能体的决策无需考虑其他智能体的利益，尽管一个智能体的动作可能会在客观上帮助或者妨害其他智能体。

13.2 多智能体系统的基本概念

本书第 3 章定义了单智能体系统的专业术语，比如状态、动作、奖励、策略、价值。在本节中，我们将这些定义推广到多智能体系统。在此后的章节中，我们用 m 表示智能体的数量，用上标 i 表示智能体的序号（ i 从 1 到 m ），依然用下标 t 表示时刻。

13.2.1 专业术语

本章依然用大写字母 S 表示**状态** (State) 随机变量，用小写字母 s 表示状态的观测值。注意，单个智能体未必能观测到完整状态。如果单个智能体的观测只是部分状态，我们就用 o^i 表示第 i 号智能体的不完全观测。

每个智能体都会做出**动作** (Action)。把第 i 号智能体的动作随机变量记作 A^i ，把动作的实际观测值记作 a^i 。如果不加上标 i ，则意味着所有智能体的动作的连接：

$$A = [A^1, A^2, \dots, A^m], \quad a = [a^1, a^2, \dots, a^m].$$

把第 i 号智能体的动作空间 (Action Space) 记作 \mathcal{A}^i ，它包含该智能体所有可能的动作。整个系统的动作空间是 $\mathcal{A} = \mathcal{A}^1 \times \dots \times \mathcal{A}^m$ 。两个智能体的动作空间 \mathcal{A}^i 和 \mathcal{A}^j 可能相同，也可能不同。比如在电子游戏中，有的士兵会远程攻击，而有的士兵只能近距离攻击，不同类型的士兵可以有不同的动作空间。

所有智能体都执行动作之后，环境依据**状态转移函数** (State-Transition Function) 给出下一时刻的状态。状态转移函数是个条件概率密度函数，记作

$$p(s_{t+1} | s_t; a_t) = \mathbb{P}[S_{t+1} = s_{t+1} | S_t = s_t, A_t = a_t].$$

它的意思是下一时刻状态 S_{t+1} 取决于当前时刻状态 S_t 、以及所有 m 个智能体的动作 $A_t = [A_t^1, A_t^2, \dots, A_t^m]$ 。

奖励 (Reward) 是环境反馈给智能体的数值。把第 i 号智能体的奖励随机变量记作 R^i ，把奖励的实际观测值记作 r^i 。在合作的设定下， $R^1 = R^2 = \dots = R^m$ ；在竞争的设定下， $R^1 \propto -R^2$ 。第 t 时刻的奖励 R_t^i 由状态 S_t 和所有智能体的动作 $A = [A^1, A^2, \dots, A^m]$ 共同决定。为什么一个智能体获得的奖励会取决于其他智能体的动作呢？举个例子，在足球比赛中，假如对方失误，自己进了个乌龙球；而你什么也没做，就获得了一分的奖励。

折扣回报 (Discounted Return) 也叫折扣累计奖励，它的定义类似于单智能体系统。第 i 号智能体的折扣回报是它自己的奖励的加权和：

$$U_t^i = R_t^i + \gamma \cdot R_{t+1}^i + \gamma^2 \cdot R_{t+2}^i + \gamma^3 \cdot R_{t+3}^i + \dots$$

此处的 $\gamma \in [0, 1]$ 是折扣率 (Discount Factor)。

13.2.2 策略网络

策略网络的意思是用神经网络近似策略函数。可以让每个智能体有自己的策略网络。对于**离散控制问题**，把第 i 号智能体的策略网络记作：

$$\hat{\mathbf{f}} = \pi(\cdot | s; \boldsymbol{\theta}^i).$$

策略网络的输入是状态 s , 输出是向量 $\hat{\mathbf{f}}$ 。向量 $\hat{\mathbf{f}}$ 的维度是动作空间的大小 $|\mathcal{A}^i|$, $\hat{\mathbf{f}}$ 的每个元素表示一个动作的概率。 $\hat{\mathbf{f}}$ 的元素都是正实数, 而且相加等于 1。做决策的时候, 根据 $\hat{\mathbf{f}}$ 做随机抽样, 得到动作 a^i , 第 i 号智能体执行这个动作。

对于**连续控制问题**, 即动作空间 \mathcal{A}^i 是连续集, 把第 i 号智能体的策略网络记作:

$$\mathbf{a}^i = \mu(s; \theta^i), \quad \forall i = 1, \dots, m.$$

有了这个策略网络, 第 i 号智能体就可以基于当前状态 s , 直接计算出需要执行的动作 \mathbf{a}^i 。

在上面的两种策略网络中, 每个智能体的策略网络有各自的参数: $\theta^1, \theta^2, \dots, \theta^m$ 。在有些情况下, 策略网络的角色是可以互换的, 比如同一型号无人机的功能是相同的, 那么它们的策略网络是相同的: $\theta^1 = \theta^2 = \dots = \theta^m$ 。但是在很多应用中, 策略网络不能互换。比如在足球机器人的应用中, 球员有的是负责进攻的前锋, 有的是负责防守的后卫, 还有一个守门员。它们的策略网络不能互换, 所以参数 $\theta^1, \dots, \theta^m$ 各不相同。

13.2.3 动作价值函数

上面讨论过, 第 i 号智能体在第 t 时刻得到的奖励 R_t^i 依赖于状态 S_t 、以及所有智能体的动作 $A_t = [A_t^1, \dots, A_t^m]$ 。因为(折扣)回报 U_t^i 是未来所有奖励 $R_t^i, R_{t+1}^i, \dots, R_n^i$ 之和, 所以 U_t^i 依赖于未来所有状态

$$S_t, S_{t+1}, S_{t+2}, \dots, S_n$$

与所有智能体未来的动作

$$A_t, A_{t+1}, A_{t+2}, \dots, A_n.$$

在 t 时刻, 回报 U_t^i 是个随机变量, 其随机性的来源是未来所有状态、所有智能体未来的动作。

如果用期望消掉回报 U_t^i 中的随机性, 就能得到价值函数。把 t 时刻的状态 s_t 和所有智能体的动作 $a_t = [a_t^1, \dots, a_t^m]$ 当做观测值, 用期望消掉 $t+1$ 时刻之后未知的状态和动作, 得到的结果就是**动作价值函数**(Action-Value Function):

$$Q_\pi^i(s_t, a_t) = \mathbb{E}[U_t^i | S_t = s_t, A_t = a_t].$$

此处的期望是关于这些随机变量求的:

- 未来的状态 $S_{t+1}, S_{t+2}, \dots, S_n$;
- 未来动作 $A_{t+1}, A_{t+2}, \dots, A_n$; 这里的 $A_k = [A_k^1, \dots, A_k^m]$ 是所有智能体在 k 时刻的动作。

上面的公式中关于动作 $A_k = [A_k^1, \dots, A_k^m]$ 求期望, (对于所有的 $k = t+1, \dots, n$) 要用到动作 A_k 的概率密度函数, 即所有 m 个智能体的策略的乘积:

$$\pi(A_k^1 | S_k; \theta^1) \times \pi(A_k^2 | S_k; \theta^2) \times \dots \times \pi(A_k^m | S_k; \theta^m).$$

也就是说, 第 i 个智能体的动作价值 $Q_\pi^i(s_t, a_t)$ 依赖于所有 m 个智能体的策略。

为什么第 i 号智能体的动作价值 $Q_\pi^i(s, a)$ 会依赖于其余智能体的策略呢? 这里给一个直观的解释。在足球游戏中, 假如你有个猪队友(即策略很差), 那么你未来获得不了

多少奖励，所以你的 Q_π^i 会比较小。假如把猪队友换成靠谱的队友（即策略更好），你的 Q_π^i 会变大。虽然你没有改变自己的策略，但是你的动作价值 Q_π^i 会随着队友的策略变化。

总结一下。如果系统里有 m 个智能体，那么就有 m 个动作价值函数：

$$Q_\pi^1(s, a), \quad Q_\pi^2(s, a), \quad \dots, \quad Q_\pi^m(s, a).$$

第 i 号智能体的动作价值 $Q_\pi^i(s_t, a_t)$ 并非仅仅依赖于自己当前的动作 a_t^i 与策略 $\pi(a^i | s; \theta^i)$ 。
 $Q_\pi^i(s_t, a_t)$ 依赖于其余智能体当前的动作

$$a_t = [a_t^1, a_t^2, \dots, a_t^m]$$

与所有智能体的策略

$$\pi(a^1 | s; \theta^1), \quad \pi(a^2 | s; \theta^2), \quad \dots, \quad \pi(a^m | s; \theta^m).$$

13.2.4 状态价值函数

我们在第 3 章中学过单智能体系统的状态价值函数 (State-Value Function)，记作 $V_\pi(S)$ ，并在策略学习的方法中反复用到 $V_\pi(S)$ 。它是对动作价值函数 $Q_\pi(S, A)$ 关于当前动作 A 的期望：

$$V_\pi(s) = \mathbb{E}_A [Q_\pi(s, A)] = \sum_{a \in \mathcal{A}} \pi(A | s; \theta) \cdot Q_\pi(s, a).$$

下面我们将状态价值函数的定义推广到多智能体系统。

第 i 号智能体的动作价值函数是 $Q_\pi^i(S, A)$ 。想要对 $Q_\pi^i(S, A)$ 关于 $A = [A^1, \dots, A^m]$ 求期望，需要用到 A 的概率密度函数，即所有 m 个智能体的策略的乘积：

$$\pi(A | S; \theta^1, \dots, \theta^m) \triangleq \pi(A^1 | S; \theta^1) \times \dots \times \pi(A^m | S; \theta^m).$$

状态价值函数可以写成：

$$V_\pi^i(s) = \mathbb{E}_A [Q_\pi^i(s, A)] = \sum_{a^1 \in \mathcal{A}^1} \sum_{a^2 \in \mathcal{A}^2} \dots \sum_{a^m \in \mathcal{A}^m} \pi(a | s; \theta^1, \dots, \theta^m) \cdot Q_\pi^i(s, a).$$

很显然，第 i 号智能体的状态价值 $V_\pi^i(s)$ 依赖于所有智能体的策略：

$$\pi(a^1 | s; \theta^1), \quad \pi(a^2 | s; \theta^2), \quad \dots, \quad \pi(a^m | s; \theta^m).$$

MARL 的困难之处就在于一个智能体的价值 Q_π^i 与 V_π^i 受其他智能体策略的影响。举个例子，在足球运动中，其他所有人的策略都没变化，只有一个前锋改进了自己的策略，让他自己水平更高。那么他的队友的价值会变大，而对手的价值会变小。一个智能体 i 单独改进自己的策略，未必能让自己的价值 Q_π^i 与 V_π^i 变大，因为其他智能体的策略可能已经发生了变化。

13.3 实验环境

如果你设计出一种新的 MARL 方法，你应该将其与已有的标准方法做比较，看新的方法是否有优势。下面介绍几种 MARL 的实验环境，用于评价 MARL 方法的优劣。

13.3.1 Multi-Agent Particle World

Multi-Agent Particle World 是一类简单的多智能体控制问题，其中包含很多种环境，如图 13.2 所示。这些环境由 Lowe 等人 [66] 开发，源代码公开在 GitHub 上：<https://github.com/openai/multiagent-particle-envs.git>。下面介绍图 13.2 中的四个环境。

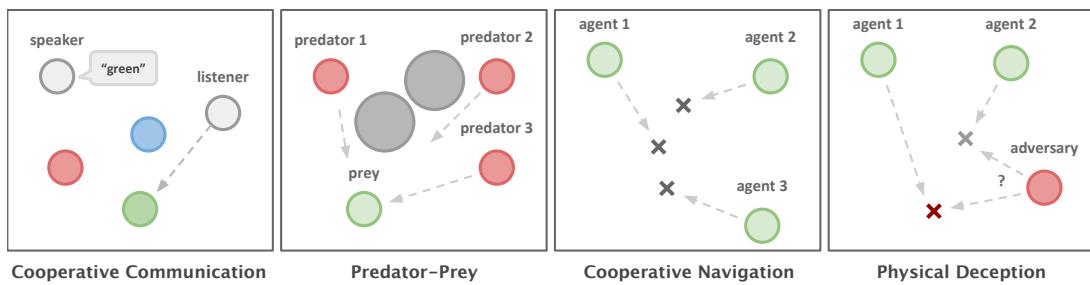


图 13.2：Multi-Agent Particle World 中的四种常用环境。图片来源于 2017 年的论文 [66]。

Cooperative Communication 这个环境中有三个点，每个点有一种颜色，这三个点不会移动。环境中有两个合作关系的智能体，一个叫做“Speaker”，另一个叫做“Listener”，它们是合作关系。任务是给定一种颜色 c ，让 Listener 移动到这种颜色的点上；离该点越近，则奖励越大。

- Speaker 的观测是 c ，即知道任务要求的颜色是什么。
- Speaker 的动作是发送一条信息，比如向量 $[0.1, 0.9, 0]$ 。很显然，训练 Speaker 的目的是让它发送的信息是颜色 c 的编码。
- Listener 的观测是三个点的颜色、Listener 与三个点的相对位置、以及 Speaker 发送的信息。比如，这是 Listener 的一个观测：

$$\left(\underbrace{[-1.5, -0.5]}_{\text{与红点的相对位置}}, \underbrace{[-0.9, -0.9]}_{\text{与绿点的相对位置}}, \underbrace{[-0.8, -0.2]}_{\text{与蓝点的相对位置}}, \underbrace{[0, 1, 0]}_{\text{Speaker 发送的信息}} \right).$$

- Listener 的动作空间是这个离散集合：{不动, 上, 下, 左, 右}。

Predator-Prey 这个环境中多个智能体，它们分为两类——多个 Predators（捕食者）与一个 Prey（猎物）。这个问题属于混合关系，即同时存在合作与竞争关系。Predators 数量多，占有优势；为了平衡双方实力，环境的设置让 Predators 速度慢于 Prey。环境中无障碍物，智能体必须绕路。

- **奖励：**如果一个 Predator 碰到 Prey（猎物），所有的 Predators 都会收到奖励，而 Prey 受到惩罚。
- **观测：**每个智能体都能观测到 (1) 它与障碍物的相对位置、(2) 它与其余智能体的相

对位置。

- **动作:** 每个智能体的动作空间都是 {不动, 上, 下, 左, 右}。

Cooperative Navigation 环境中有 m 个合作关系的智能体与 m 个不动的点。

- **奖励:** 每个不动点都带有奖励, 离该点最近的智能体会收集到奖励, 奖励的大小与距离负相关。也就是说, 最好的策略是让 m 个智能体分别覆盖 m 个点。智能体应当远离彼此; 如果两个智能体碰撞, 则会受到惩罚。
- **观测:** 每个智能体都能观测到与其他智能体的相对位置、以及与 m 个点的相对位置。
- **动作:** 每个智能体的动作空间都是 {不动, 上, 下, 左, 右}。

Physical Deception 这个环境中 $m+1$ 个智能体, 其中 m 个是合作关系的玩家, 一个是对手。这个问题属于混合关系。

- **奖励:** 环境中有 m 个点, 其中一个点 x 带有奖励, 离 x 距离最近的玩家获得奖励, 奖励的大小与距离负相关。也就是说, 应当有一个玩家到达点 x ; 但这是不够的。对手也想到达点 x ; 对手离 x 越近, 对手得到的奖励越大, 而对手的奖励是玩家的惩罚。
- **玩家的观测:** 玩家知道它与点 x 的相对位置、与其余 $m-1$ 个点的相对位置、与其余 $m-1$ 个玩家的相对位置。
- **对手的观测:** 对手知道与所有 m 个点的相对位置、与所有 m 个玩家的相对位置, 但是不知道 m 个点中哪个点是 x 。
- **动作:** 每个智能体的动作空间都是 {不动, 上, 下, 左, 右}。

虽然只有当覆盖 x 的时候有奖励, 玩家也不能仅仅覆盖点 x , 而不覆盖其余的点。否则对手会推测出 x 是哪一个点。因此, 玩家最好的策略是覆盖所有 m 个点, 从而迷惑对手。



(a) 双方各有 3 只 Stalkers 和 5 只 Zealots



(b) 一方有 7 只 Zealots, 另一方有 32 只 Banelings

图 13.3: SMAC 库中的两种环境。

13.3.2 StarCraft Multi-Agent Challenge (SMAC)

星际争霸 2 (StarCraft II) 是由暴雪在 2010 年推出的一款即时战略游戏。游戏中有很多兵种 (即很多类型的智能体), 每个兵种有自己的生命值、护甲、移动速度、攻击范围、

杀伤力等属性。每个玩家控制多个士兵，一个士兵可以攻击对手的士兵。一个士兵在生命值耗尽的时候死去，从游戏中消失。

StarCraft Multi-Agent Challenge (SMAC) 是基于星际争霸 2 开发的库，用来评价 MARL 方法的好坏。SMAC 由 Samvelyan 等人 [83] 开发，源代码公开在 GitHub 上：<https://github.com/oxwhirl/smac.git>。SMAC 库中有很多对战的环境。图 13.3 展示了两种环境。

SMAC 中每个士兵是一个智能体，有自己的观测（多个向量），能做出离散动作。如图 13.4 所示，每个士兵有自己的视野，能观测到一个圆内的所有队友和对手。每个士兵的观测表示为一个列表，列表的每个元素是一个向量，一个向量对应视野范围内的一名士兵（队友或对手）。每个向量包含以下信息：

- 距离、相对位置、生命值 (Health)、护甲 (Shield)、士兵类型 (Unit Type)、上一个动作（仅知道队友的上一个动作）。

动作空间是离散的，每个士兵每次可以执行下面动作中的一种：

- 向东、西、南、北四个方向中的一个移动。
- 攻击对手（或治疗队友），仅限攻击范围之内，需要指定被攻击（或被治疗）目标的 ID。
- 停止攻击。

一个团队的士兵是合作关系，奖励是给予团队的，而不是给具体某个士兵。SMAC 有两种类型的奖励可供选择。一种是稀疏的奖励：最终胜利获得奖励 +1，失败获得奖励 -1；胜利的意思是杀死对方所有士兵，己方至少有一个士兵存活。另一种是稠密的奖励：杀死对方一个士兵有正奖励，己方士兵被杀有负奖励；胜利获得正奖励，失败获得负奖励。



图 13.4：红色虚线是士兵的攻击范围，青色实线是士兵的视野。图片来自论文 [83]。

13.3.3 Hanabi Challenge

花火 (Hanabi) 是一种合作型的卡牌游戏，玩家不能观看自己的牌，只能看其他玩家的。游戏的玩法是要将不同花色的数字牌按顺序排列。每回合中玩家只能获得有限的信息，需要做推理，从而做出决策。花火的规则较为复杂，此处不详细解释。有兴趣的读者可以在互联网上检索“花火卡牌游戏”，了解游戏规则。Hanabi Challenge 由 Bard 等人 [9] 在 2020 年开发，源代码公开在 GitHub 上：<https://github.com/deepmind/hanabi-learning-environment.git>。该程序提供花火游戏的环境，可供 MARL 学术研究。

从强化学习的角度来看，花火属于合作类型的 MARL。一局游戏结束时有奖励，奖

励是给予团队的，而非玩家个人。每个玩家相当于一个智能体，他无法观测到全局状态，只能在不完全观测的情况下做出决策。玩家可以看到其他玩家的牌，但是不能看自己的牌；玩家要靠其他玩家提供的情报来推测自己的牌。玩家每一回合可以做出三种动作中的一种：提供情报、弃置一张牌、打出一张牌。提供情报的次数是很有限制的，玩家必须学会传递最有用的情报。出牌的好坏会影响奖励。

相关文献

合作关系的 MARL 在自动控制领域被称作 Team Markov Games [49, 113, 123]。合作关系的 MARL 在 AI 领域最早见于论文 [17, 58]。竞争关系的 MARL 最早见于论文 [64]。混合关系的 MARL 最早见于论文 [52, 57, 65]。

很早就有关于将 Q 学习等价值学习方法推广到 MARL 的研究。1993 年的论文 [102] 研究了独立 Q 学习 (Independent Q-Learning, 缩写 IQL)，即智能体独立做 Q 学习，不共享信息。2017 年的论文 [39, 101] 将 IQL 用在深度强化学习。比较有名的多智能体价值学习方法有 Value-Decomposition Networks [95]、QMIX [80] 等方法。目前 MARL 更流行 Actor-Critic 方法，比如 [43, 38, 66, 53]。其中最有名的是 2018 年的 COMA [38] 与 2017 年的 MADDPG [66]。

对 MARL 感兴趣的读者可以阅读这些综述和书籍：Weiss 1999 [117]，Stone & Veloso 2000 [94]，Vlassis 2007 [112]，Shoham & Leyton-Brown 2008 [90]，Buşoniu *et al.* 2010 [23]，Zhang *et al.* 2019 [125]。