arXiv:2602.00951v1 [cs.AI] 1 Feb 2026

R-HTN：面向安全与游戏人工智能的叛逆在线HTN规划

Héctor Muñoz-Avila HEM4@LEHIGH.EDU
理海大学计算机科学与工程系；美国宾夕法尼亚州伯利恒市，邮编 18015-3084

DavidW. Aha DAVID.W.AHA.CIV@US.NAVY.MIL 海军人工智能应用研究中心；美国海军研究实验室；美国华盛顿特区

Paola Rizzo P.RIZZO@INTERAGENS.COM
Interagens s.r.l.；意大利罗马

摘要

我们介绍一种在线分层任务网络（HTN）智能体，其行为受一组内置指令 $\mathcal{D}$ 支配。与其他能够反叛（即智能抗命）的智能体一样，在某些情况下，我们的智能体不会执行用户分配的任务，而是采取不符合用户预期的行动。我们的工作结合了三个概念：HTN规划、在线规划，以及在执行用户分配的任务时必须考虑的指令 $\mathcal{D}$。我们研究了两种智能体变体：(1) 一种非自适应智能体，如果发现自身违反了 $\mathcal{D}$，就会停止执行；(2) 一种自适应智能体，在同样的情况下会修改其HTN计划，寻找完成给定任务的替代方法。我们提出了R-HTN（即反叛型HTN），这是一种在指令 $\mathcal{D}$约束下进行在线HTN规划的通用算法。我们在两个任务域中评估R-HTN，在这些领域中，智能体出于安全原因或其人格特质的要求，不得违反某些指令。我们发现，R-HTN智能体从不违反指令；只要可行，它们就会努力实现用户给定的目标，但不一定会按照用户预期的方式实现。

1. 引言

随着自主智能体日益普及，用户指示其执行不安全甚至危险行为的频率也在增加。这可能是有意为之。例如，一些新闻报道经常提到，无人机（UAVs）会违反保留空域或机场等“红色区域”（Huang，2025）的规定。

有时，敌对行为者可能会诱使用户在指挥智能体进行监控时犯错（例如，疏忽某些事项）。例如，行为者可能会将用户的注意焦点引向无关事件，从而导致用户减少对其智能体的关注。更广泛地说，不安全地使用自主智能体可能是认知超载等因素造成的。当用户同时执行多项任务（例如控制多个自主智能体），并发现自己无法监控智能体正在执行的某些动作时，就可能发生这种情况。事实上，在 Gartenberg等人（2014）报告的实验中，负责控制多架自主无人机的用户经常未能在无人机飞越意外受到限制的空域时改变其航线（例如，检测到携带反无人机武器的敌对智能体所在的空域）。在这些实验中，用户需要重新为无人机规划航线，以

© 20XX 认知系统基金会。保留所有权利。

避开危险区域。用户未能为无人机重新规划航线，部分原因在于他们需要（1）专注于其他任务（例如，指挥智能体调整位置以拍摄航拍照片），以及（2）同时控制多架无人机的导航。

受这些因素的启发，我们研究赋予在线HTN规划智能体反抗能力的效用。我们假设，分层任务网络规划域 Σ 配有一组指令 $\mathcal{D}$（例如“避开任何红色区域”）和任务修正程序。如果智能体发现自己处于状态 s，且存在偏差（即 $\delta(s) = True$），那么它可以根据分配给自己的任务 t、Σ 以及任务修正程序，尝试修正待执行的任务。在这种情况下，我们区分两种类型的反抗型HTN（R-HTN）智能体：

- **非自适应智能体**：每当智能体发现自己违反了$\mathcal{D}$ 时，就会停止执行任务 t，并等待用户发出新指令。
- **自适应智能体**：每当智能体发现自己违反了 $\mathcal{D}$，就会尝试执行纠正动作（如果可行）。[1]

R-HTN的设计遵循以下智能体设计原则：

- **分层任务网络规划**。一种分层规划范式，其中任务（例如*achieve(g)*，即实现用户给定的目标 g）会被递归分解为更简单的任务，直至生成所谓的原始任务（对应于动作）；执行这些动作后，便可实现 g。

- **指令**。指令是一个函数 δ：$S \rightarrow \{True, False\}$，对于给定的状态 $s \in S$，若 s 是一个意外状态（也称为**-偏差**），需要智能体加以关注（例如，智能体发现自己身处禁止进入的危险（“红色”）区域），则该函数返回真。

- **在线分层任务网络规划**。智能体交替进行分层任务网络规划与执行；随着智能体访问新的状态，它会检测是否存在偏差，并可能据此采取行动。

我们对基于分层任务网络的叛逆型智能体感兴趣，这源于相关报告指出，分层任务网络尤其适用于许多任务，包括军事规划（Donaldson，2014）、战略决策（例如游戏中的决策（Smith等人，1998；Verweij，2007））以及控制多个智能体（Cardoso与Bordini，2017），其中也包括无人机团队（Musliner与Goldman，2010）。R-HTN智能体具有通用性，只要给定 Σ、$\mathcal{D}$和任务修复程序，便可用于任何任务领域。

在第2节中，我们描述了一个示例场景，用于说明R-HTN智能体的预期行为。接下来，我们介绍一种 $\mathcal{D}$差异分类体系（第3节）。随后，我们讨论在线分层任务网络规划（第4节），并介绍R-HTN智能体算法（第5节）。然后，我们通过两个领域开展一项实证研究，并讨论研究结果（第6节）。最后，我们讨论相关工作（第7节），并给出总结性说明（第8节）。

1. 这种情况不可行的一个例子是：智能体已经执行了为实现 t所分配的最大动作数量。

2. 示例场景

我们的第一个场景受RESCHU任务领域的启发（Boussemart & Cummings, 2008）。RESCHU（异构无人飞行器监督控制研究环境）旨在研究认知超载如何影响用户的决策。RESCHU用户指挥一组无人机前往目的地；他们还必须确保这些无人机避开红色区域（危险区域）。随着时间推移，每个红色区域都会消失，随后随机出现在另一个位置（该位置不包含任何智能体）。在我们的场景中，我们将其称为O-RESCHU（即：自有RESCHU），所有智能体都从同一位置开始，并拥有相同数量的能量点。智能体移动到相邻单元格时会消耗一个能量点。用户为每架无人机分配一个目标位置，该位置应避开所有红色区域（闯入红色区域以及在红色区域内移动都会导致无人机损失更多能量点）。

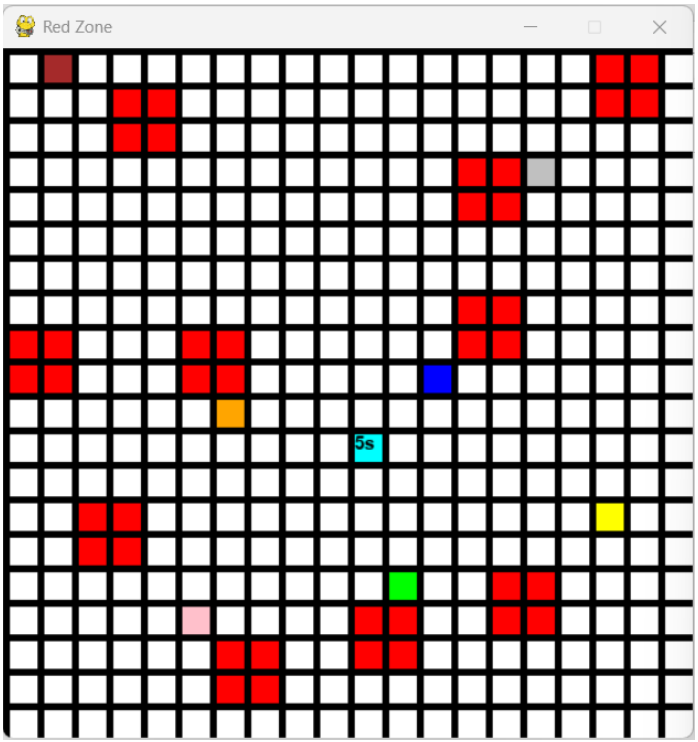


图 1　：一张随机生成的地图，五个智能体均从标记为“5s”的位置出发

用户的目标是让其智能体访问指定的目的地，同时尽量减少消耗的能量点数。用户希望每个智能体从当前位置沿最短路径前往用户为其指定的目标位置，但不会事先获知红色区域何时、何地会消失和重新出现。红色区域可以理解为会导致无人机消耗比平时更多能量的气象条件，因此用户应确保其无人机避开这些区域。

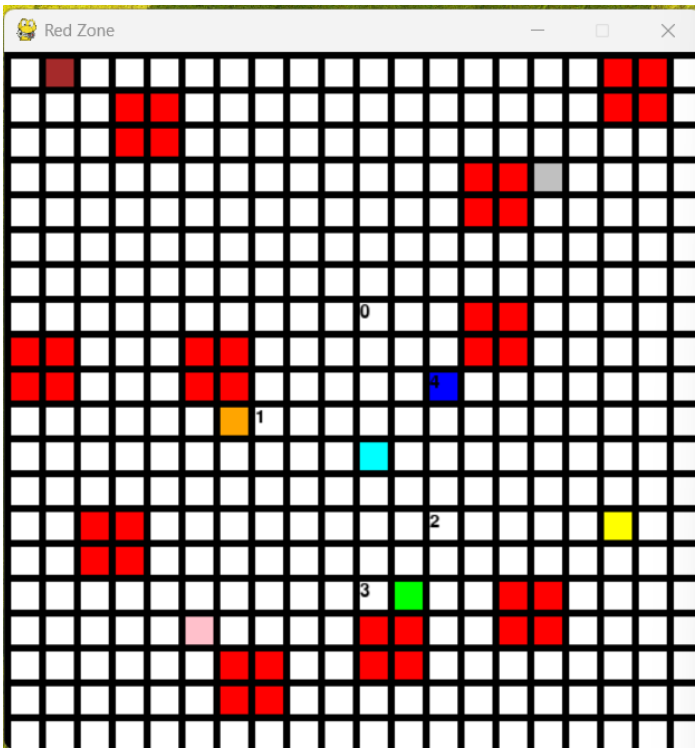


图2：经过四个时间刻后五个智能体的位置。

图 1 展示了 O-RESCHU 随机生成配置的一个快照。10 个红色正方形（组）是智能体应避开的红色区域（我们将其描绘为由四个相邻单元格组成的正方形）。标有“5s”的彩色单元格位置是五个智能体的共同起始位置。其他七个彩色单元格是目的地。

图2展示了经过四个时间刻后的快照（每个时间刻中，每个智能体最多移动到一个相邻单元格——上方、下方、左侧或右侧）。智能体4已抵达其（深蓝色）目的地。智能体3和智能体1将在下一个时间刻分别抵达其（绿色）和（橙色）目的地。智能体0正朝棕色位置（左上方）移动。最后，智能体2正朝其（黄色）目的地移动。

3. 分类体系

我们考虑两种类型的智能体：在推理过程中忽略 $\mathcal{D}$ 的智能体，以及重视 $\mathcal{D}$ 的智能体。那些试图完成用户分配的任务而不考虑其他因素的智能体称为**遵从型** 智能体（Coman & Muñoz-Avila，2014）。在我们的工作中，遵从型智能体忽略$\mathcal{D}$，因此仅依据分层任务网络领域 Σ生成并执行计划。在重视 $\mathcal{D}$ 的智能体中，我们根据它们应对 $\mathcal{D}$差异的方式加以区分（见表1）。

第一行用于让智能体测试当前状态中是否存在偏差 s（我们将其称为一个 $\mathcal{D}$-偏差），例如某个智能体发现自己位于红色区域单元格中。令 s 为一组

表1： $\mathcal{D}$偏差的分类体系

Type	示例	形式化描述
即时 $\mathcal{D}$偏差	位于红色区域	δ（s） $= True$
预测的 $\mathcal{D}$-偏差	将达到 红色区域	$\delta(\pi_n(s)) = True$
Adaptive $\mathcal{D}$-偏差	转向以避开 红色区域	$\delta(\pi'_n(s)) = False$ and $\pi'(s) \vDash g$

接地谓词。例如， $s = \{at$（5，（10，11））red（7，（10，11），2）$\}$表示智能体 5 位于单元格（10，11）中。它还表示编号为 7 的红色区域位于（10，11），且该红色区域的大小为 2×2。令 $\mathcal{D}$为包含 10 条指令的集合， δ_0，.....， δ_9，每个红色区域对应一条指令。如果δ_i（s） $= True$，则表示至少有一个智能体在状态 s中发现自己位于红色区域 i 内。因此，在这种情况下， δ_7（s） $= True$，因为智能体 5 位于红色区域 7 内。

表1的第二行适用于能够预测未来状态中是否会出现 $\mathcal{D}$-偏差的智能体。我们将其称为预测的 $\mathcal{D}$-偏差。再次令 s 为一组接地谓词。令 π_n =($a_1 a_2 \ldots a_n$)表示智能体接下来要执行的动作 n。令 π_n(s)表示从状态 s开始执行接下来的动作 n 后得到的状态:

$$\pi_n(s) = \begin{cases} a_1(s) & \text{if } n = 1 \\ a_n(s') & \text{if } n > 1 \text{ and } s' = \pi_{n-1}(s) \end{cases} \tag{1}$$

在这种情况下，如果对于 $n \geq 1$， $\delta(\pi_n(s)) = True$，则会出现预测的 $\mathcal{D}$-偏差。

第三种情况，*Adaptive* $\mathcal{D}$偏差，指的是智能体生成一个计划 π'（即对计划 π 的修复），使其在执行接下来的 n动作时不会出现预测的 $\mathcal{D}$偏差，并且π' 实现用户给定的目标 g。这一情况在表1第3行的最后一列中形式化定义。

4. 分层任务网络规划

分层任务网络规划是一种规划范式，其中复杂任务被分解为更简单的任务，直至生成一系列所谓的原始任务（Georgievski & Aiello, 2015）。每个原始任务都通过一个动作来实现；因此，HTN规划的输出是一系列动作 π。我们用 T 表示所有原始任务和非原始任务的集合。HTN 规划模型 Σ 是由动作 和方法构成的集合，我们接下来将对此进行定义。

动作 a 定义了通常的状态转移函数 a: $S \to S \cup \emptyset$，其中 S 是所有状态的集合。如果 a(s) $= \emptyset$，那么 a在 $s \in S$ 中不适用。否则， a(s) $\in S$ 是执行动作 a 于 s 后得到的状态。

方法 m 用于分解非原始任务，其中 T 如果 $m(s,t) = \emptyset$，则 m 在$\tilde{}$(s, t)中不适用。否则， m(s, t) $\in T$是一个任务序列。在这种情况下，我们说 t 被分解为m(s, t)。

为了举例说明一种方法远距离导航，设 s 为图2中所示的状态。远距离导航用于完成到达（智能体，位置），每当目标位置距离智能体至少2个格子时。在此状态中，智能体0当前位于位置（x, y）。该智能体已被指派前往棕色目标位置（靠近左上角）。对于这一特定状态，远距离导航方法将非原始任务*reach(0,brown)* 转换为两个子任务。第一个是原始任务*up(0)*，它执行将智能体0移动到（$x, y-1$）的动作；第二个任务是对非原始任务*reach(0,brown)*的递归调用。简而言之，远距离导航*(reach(0,brown))* = *(up(0),reach(0,brown))*。

为了举例说明另一种方法——近距离导航，请考虑智能体3，它被指定以绿色位置为目标。也就是说，智能体3必须完成非原始任务*reach(3,green)*。在这种情况下，近距离导航方法会将该任务分解为一个原始任务*right(3)*，执行该任务后，智能体3便会移动到其目标位置。简而言之，*navigate-close(s,reach(3,green))* =*(right(3))*。

总而言之，我们描述了两种方法：一种用于智能体的目的地位于智能体位置上方且至少相距两个格子的情况，另一种用于智能体的目的地与其当前位置相邻的情况。但反过来则不成立。也就是说，当智能体与目标位置相邻时，远距离导航方法不适用于这些状态。同样地，当智能体与目标位置不相邻时，近距离导航也不适用。因此，

远距离导航*(s,reach(3,green))* = ∅近
距离导航*(s,reach(0,brown))* = ∅

5. 叛逆在线HTN规划

算法1展示了R-HTN的伪代码。该伪代码基于SHOP系统中的HTN规划（Nau等人，1999年）。未加下划线的部分是标准的SHOP伪代码，加下划线的部分则是我们新增的内容。

R-HTN规划过程 R-HTN(s, $\tilde{t}$) 接收一个状态 s 和一个任务列表 $\tilde{t} \in \tilde{T}$作为输入（第1行）。它使用与R-HTN相同的参数，并附加一个空列表，调用RSeekPlan。该空列表表示空计划（即不包含任何动作的计划）。RSeekPlan通过递归分解 $\tilde{t}$，按如下方式生成解计划 π：如果任务列表 $\tilde{t}$ 为空（即 ()），则返回空计划（第6行）。否则，$\tilde{t} = (t_0, t_1, \ldots, t_n)$ 是一个非空任务列表（第 7 行）。接下来分为两种情况：

- (情况1) 如果 t_0 是原语，且其关联动作 a_0 在 s 中适用（即 $a_0(s) \neq \emptyset$；第~行），则检查是否需要修复，并返回任务列表 t'（第11行；稍后我们将对此进行详细说明）。如果 t 与 t'不同，则 t' 修复 t，并在 t' 上递归继续规划（第12和13行），同时保持 π 和 s 不变。否则，执行 a_0，得到新状态 s'（第14行）。然后，使用新状态 s'、剩余任务列表（t_1，……，t_n）以及添加了 a_0 的计划 π 递归继续规划（第15行）。

- (情况2) 如果 t_0 是非基本任务，且存在一个方法 m 适用于 s 和 t_0（即 $m(s, t_0) \neq \emptyset$；第16–18 行），则使用相同的状态 s 和计划 π 继续递归规划，并

算法1 R-HTN算法

1: **过程**R-HTN(s, $\tilde{t}$)
2: **返回** RSeekPlan(s, $\tilde{t}$, 0) ▷ () 是空计划；一个不包含任何动作的计划
3: **结束过程**
4:
5: **过程**检索计划(s, t, π) ▷ π 是目前为止生成的计划
6: **if** $\tilde{t} = ()$ **则返回**() ▷返回空计划
7: let $\tilde{t} = (t_0, t_1, ..., t_n)$
8: **如果** t_0 是原语**则**
9: 令 a_0 为与 t_0关联的动作
10: **如果** $a_0(s) = \emptyset$**则返回** $\emptyset$
11: $\tilde{t'} \leftarrow$RepairTasksIfNeeded($\tilde{t}, s, a_0$)
12: **if** $\tilde{t'} \neq \tilde{t}$ **然后** ▷ 发生 D-差异
13: **返回**RSEEKPLAN(s, $\tilde{t'}, \pi$) ▷ 递归调用
14: $s' \leftarrow$ **执行** a_0 上 s
15: **返回** RSEEKPLAN(s', (t_1, ..., t_n) $\pi \cdot (a_0)$) ▷ 递归调用
16: **如果** t_0 是复合项**则**
17: **对** $m_0 \in M$ **执行** ▷ M 是所有方法的列表
18: **如果** $m_0(s, t_0) \neq \emptyset$ **则**
19: let π' = RSEEKPLAN(s, $m_0(s, t_0) \cdot (t_1, ... t_n)$ π)
20: **如果** $\pi' \neq \emptyset$, **则**
21: **返回** π'
22: **返回** $\emptyset$
23: **结束过程**
24:
25: **过程**REPAIRTASKSIFNEEDED($\tilde{t}, s, a_0$)
26: **对** $\delta \in \mathcal{D}$ **执行**
27: 如果 $\delta(s)$，则 ▷ δ 在状态 s中被违反
28: **返回** repairTaskListState(δ,s, $\tilde{t}$) ▷ 返回一个更新后的列表
29: **对** $\delta \in \mathcal{D}$ **执行** ▷ 仅在当前状态没有违规时进行检查 s
30: 如果 $\delta(a_0(s))$ 那么 ▷ δ 在状态 $a(s)$ 中被违反
31: **返回** repairTaskListEffect(δ,s, $\tilde{t}$,a_0) ▷ 返回更新后的列表
32: **返回** $\tilde{t}$ ▷ 未违反任何指令；原样返回任务列表
33: **结束过程**

增强任务列表 $m(s, t_0) \bullet (t_1, ..., t_n)$，其中 $\bullet$ 表示任务列表的串接。[2] 如果递归调用返回一个非空计划 π'，则返回该计划（第 21 行）。

2. 即，$(t_0, ..., t_n) \bullet (t'_0, ..., t'_m) = (t_0, ..., t_n, t'_0, ..., t'_m)$

第22行是兜底情况： t_0 要么是原语项，但情况1没有得出解；要么是复合项，但情况2没有得出解。在这两种情况下，该过程都会返回 $\emptyset$，表示不存在这样的计划。

在线分层任务网络规划 我们关注动态环境，在这种环境中，世界状态会因智能体自身的动作以及独立于智能体自身动作的因素而发生变化。例如，某个区域可能开始下雨（因此变成红色区域），也可能另一个区域停止下雨（因此不再是红色区域）。为了恰当地应对这些情况，R-HTN 在第14行使用在线规划。它执行选定的动作 a_0，并返回在环境中执行 a_0 后观察到的状态 s'（第6节将介绍这两个领域的模拟器）。

任务修复程序。 第25至33行详细说明了 RepairTasksIfNeeded 过程。它接收˜当前任务列表 t、当前状态 s 以及将要执行的动作 a_0（第25行）。首先检查当前状态 s 中是否存在$\mathcal{D}$偏差（第26和27行）。如果存在，则调用领域专用过程 repairTaskListState，该过程会返回更新后的任务列表（第28行）。如果当前状态中不存在 $\mathcal{D}$偏差，则检查投影状态 δ中是否存在任何 $\mathcal{D}$偏差 $a_0(s)$（第29和30行）。如果存在，则调用领域专用过程 repairTaskListEffect，该过程会返回更新后的任务列表。调用 $a_0(s)$不会改变世界状态。这只是通过标准规划来计算投影的下一状态。在 R-HTN 中，只有第14行会在环境中执行动作。第6节将介绍我们针对两个任务域设计的任务修复程序。

6. 实证评估

我们进行了评估，测试了前文所述的三种在线分层任务网络规划智能体：Compliant、Nonadaptive 和 Adaptive。在这些实验中，模拟用户随机选择每个智能体要访问的目的地。

我们的假设如下：

- 与 Nonadaptive 和 Compliant 相比，Adaptive 将实现更多目标。
- Adaptive 和 Nonadaptive 不会出现任何状态违规。

我们有两个领域：

- **O-RESCHU**。本文迄今讨论的领域。模拟用户从不将两个或更多智能体分配到同一位置。

- **MONSTER**。为了让非玩家角色更加逼真，游戏设计师提出了非玩家角色阵营的概念（Gygax，1978）。非玩家角色阵营指的是“善良”或“邪恶”等人格特质，其理念是：如果玩家命令某个NPC执行违背其阵营的动作（例如，命令一个具有“善良”阵营的NPC偷窃金币），该NPC可能会拒绝执行命令。MONSTER是对我们的O-RESCHU模拟所做的一项直接修改：红色区域被重新解释为怪物，彩色地点被重新解释为黄金地点，每个地点都有5枚金币。（模拟的）用户将NPC代理派往这些地点收集

金币。按照许多游戏的惯例，NPC代理通过移动到有金币的单元格来收集金币（即不存在用于收集金币的显式动作）。金币被收集后，该地点将不再有金币。NPC和怪物开始时各有10点生命值（hp）。当怪物和NPC处于同一单元格时，它们会“战斗至死”：具体而言，我们会反复掷骰子，每次有50%的概率使怪物损失1点生命值，另有50%的概率使NPC损失1点生命值。当任一方的生命值降至0时，该方将从游戏中被淘汰。如果NPC的生命值降至0，模拟将终止，因为每场游戏中只有一个NPC在行动。如果怪物的生命值降至0，怪物将从游戏中消失。NPC会以战斗后剩余的生命值继续行动。这意味着，平均而言，一个NPC无法撑过与怪物的一场战斗；即使能够存活，通常也只剩下很少的生命值，因此不太可能赢得第二场战斗。怪物不会移动，但可以在其他地点重生，不过绝不会重生在NPC所在的单元格中。它们可以在黄金地点重生。

6.1 任务修复

O-RESCHU. 重新考虑图2中的场景，假设智能体3（它与绿色目标位置相邻）被改为分配去导航至粉色位置（即其修订后的目标）。再次假设，$\mathcal{D}$ 中唯一的指令是让该智能体避开任何红色区域，并且其当前任务列表为：*(down(3), reach(3,pink))*。

在这种情况下，将会出现与 $n = 1$的预计 $\mathcal{D}$-偏差（即，应用 π_1 =($down$(3))会导致 $\mathcal{D}$-偏差）。为应对这种情况，我们的智能体将通过考虑当前状态 s 和目标 g来修复任务列表。我们用 $A(s)$表示在 s 中可应用、且应用它们不会产生 $\mathcal{D}$-偏差的备选动作。在这种情况下，$A(s)$ =$\{up(3)\ left(3)\ right(3)\}$。所选的备选动作 a 是满足以下条件的一个：

$$a = min_{a \in A(s)} dist(a(s), g), \tag{2}$$

其中 $dist$ 是距离函数（本例中我们使用曼哈顿距离），而*min* 是最小值。

在此场景中，之所以选择 $a = left(3)$，是因为它与另外两个备选方案相比，能将智能体3移动到距离粉色目的地最近的位置。因此，智能体3将通过用 $left(3)$替换 $down(3)$来修改任务列表。由此得到修改后的任务列表：*(left(3), reach(3,pink))*。随后，我们的智能体将继续执行在线分层任务网络规划过程，将智能体3向左移动一次，然后使用*reach(3,pink)*递归地执行在线规划。

由于该智能体正在进行在线规划，它将恰当地应对新的突发情况，例如当红色区域出现在其路径上时。类似地，如果导致任务修复的红色区域消失，则该智能体将考虑通往其目标的新的可行路径。

在我们的场景中，智能体不会被红色区域“困住”。红色区域始终是正方形，会随机放置在地图上，但彼此之间留有间隙。这意味着智能体总能在红色区域之间穿行，到达任何位置。在其他智能体可能被困住的任务域中，无论它采取哪种替代方案，$A(s)$ 都将定义为可行规划中所有用于到达 g的首个动作的集合。如果 $A(s)$ 为空，那么我们的智能体将停止执行。

MONSTER。 它只有一条指令，即 $\delta_{monster}(s)$；当它发现自己与怪物位于同一个单元格时，该指令为真。由于智能体会检查下一个动作是否会导致 $\mathcal{D}$偏差（算法第 30 行），而怪物不会出现在 NPC 所在的同一个单元格中，这意味着我们的自适应NPC智能体能够提前知道自己在沿路收集金币时是否会遇到怪物。我们的自适应NPC智能体表现出懦弱特质，因为它会避开与怪物战斗。也就是说，它会以与 O-RESCHU 中任务修复完全相同的方式修改任务列表（即绕开怪物）。如果指定的金子位置有怪物，智能体就会放弃用户指定的任务，从而发生反叛，但这符合其懦弱的性格特质。

6.2 任务领域地图与设置。

O-RESCHU。 我们的地图是由 20×20 个格子组成的二维网格。其中包含 10 个大小为 2×2 的红色区域、 7 个（彩色）目的地（即格子），以及供全部 5 个智能体共用的 1 个初始起始位置。这些设置如图 1 和图 2 所示。目的地和起始位置会在每个回合开始时随机生成，并在整个回合期间保持不变。红色区域会以固定概率重生。在实验中，我们使用以下概率：0（即红色区域在整个回合期间保持不变）、5、10、15、20、25、30、35、40、45 和 50（即红色区域有 50% 的概率重新定位，因此平均每两个时间刻重新定位一次）。红色区域不会生成在智能体上方，但可能会生成在目的地上方。智能体不能斜向移动，也可以留在当前位置不动。

MONSTER. 地图的尺寸和红色区域（此处为怪物）的数量与 O-RESCHU 相同，彩色地点（包含金币）亦然。怪物以与 O-RESCHU 相同的概率设置生成。唯一的区别是，网格中只有一个智能体行动。

对于这两个领域，针对每种红色区域（或怪物）重生概率以及 3 个智能体中的每一个，我们都运行了 100 回合，并计算了因变量测量的平均值，接下来将对此进行说明。

6.3 因变量测量。

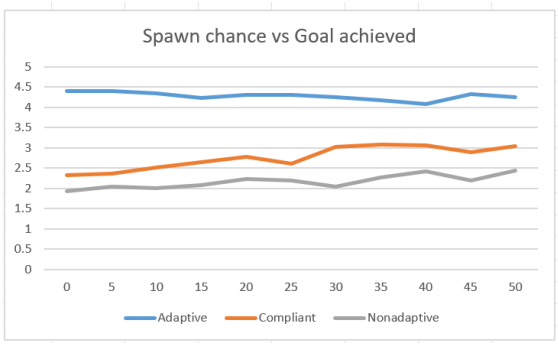
两个领域的共同测量。 第一项是二元测量，表示智能体是否达成目标（即到达其指定位置）。第二项是一个计数器，用于统计智能体遭遇的 $\mathcal{D}$ 次差异。

O-RESCHU. 对于这一任务领域，我们还测量智能体获得的罚分。智能体每次在非红色单元格内移动一步，罚 1 分；每次移动一步进入、离开或穿过红色单元格，则罚 20 分。在每个回合开始时，每个智能体会获得 38 分，这代表沿最短路径在地图上任意两个地点之间移动所需的最大步数。当智能体的剩余分数降至 0 时，它便会在该回合中停止移动。

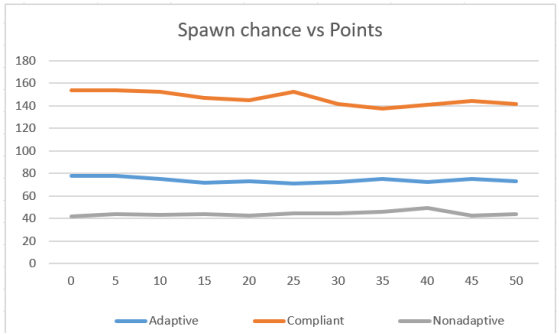
MONSTER. 在每个回合中，模拟用户会指定一个金子位置，并在智能体到达第一个位置或放弃尝试到达该位置后，再指定第二个位置。对于每一场游戏，我们计算两项额外的因变量测量。第一项是收集的黄金总量，这是一个介于0到35之间的数字（例如，如果黄金地点沿着通往其

目的地的最短路径聚集在一起，从而使智能体将它们全部访问）。第二项测量是智能体死亡数（0或1）。

6.4 O-RESCHU 的结果



(a) 横轴表示红色区域生成的概率（每个时间刻度）。纵轴表示每种智能体类型在100个回合中平均达成的目标数量。



(b) 横轴表示红色区域生成的概率（每个时间刻度）。纵轴表示每种智能体类型在100个回合中平均产生的罚分。

图3：性能指标随红色区域生成概率的变化。

O-RESCHU 的结果如图3a、图3b和图4所示。每幅图的横轴表示每个时间刻度中红色区域重生的概率（从0%到50%）。为考虑随机性，每个数据点均为100次运行的平均值。

实验结果 - 达成的目标：如图3a所示，Adaptive 智能体每回合达成的目标平均数量最高，介于 4 到 4.5 之间。目标数量的最大值为 5（即每个智能体一个）。Adaptive 智能体平均达不到 5 个目标有两个原因。首先，红色区域有时会出现在指定的目标位置上，导致相关智能体无法到达那里（即避免违反 $\mathcal{D}$）。其次，智能体在绕开红色区域时可能耗尽点数。Compliant 智能体每回合平均达成 2 到 3 个目标。随着红色区域的生成频率增加，这一数量也会增加，因为红色区域有时会移出智能体前往目标位置的路径。此外，Compliant 智能体可以移动到红色区域内的目标位置。最后，Nonadaptive 智能体每回合平均达成 2 至 2.5 个目标。Nonadaptive 智能体达成的目标少于 Compliant 智能体，因为前者在移动会导致违规时总会放弃目标，而后者仍会继续移动，有时还能达成目标。本实验验证了我们的假设：Adaptive 达成的目标多于 Nonadaptive 和 Compliant。

实验结果——产生的罚分：图3b展示了平均罚分。遵从型智能体产生的平均罚分最多（140-160），几乎是自适应智能体的两倍。遵从型智能体的平均罚分略有下降，因为红色区域有时会移出其计划路径。非自适应智能体

产生的罚分较少，因为它们更频繁地放弃目标，如图3a所示。

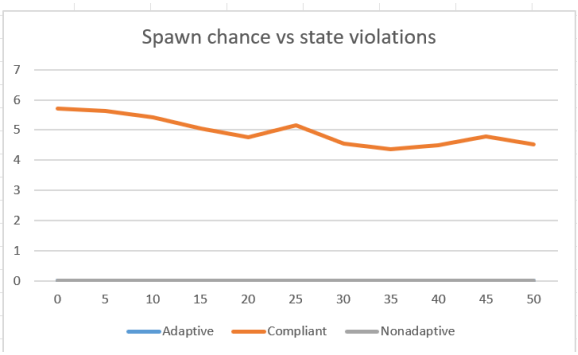


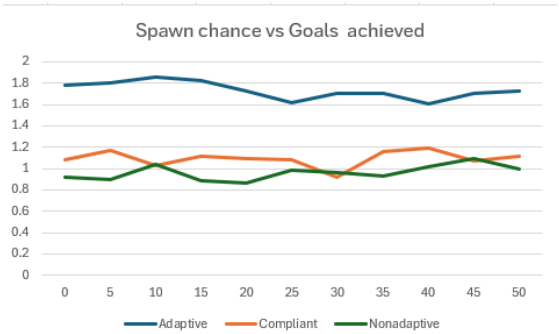
图4：横轴表示红色区域生成的概率（每次刻度变化时）。纵轴表示每种智能体类型每100回合平均发生的 $\mathcal{D}$偏差数量。

实验结果——发生的状态违规：图4展示了每种智能体类型平均发生的 $\mathcal{D}$偏差数量。自适应智能体和非自适应智能体从未发生过 $\mathcal{D}$偏差。相比之下，遵从型智能体每回合平均发生4至6次状态违规。同样，随着重生概率增加，这一数值会下降，因为红色区域生成（和消失）得越频繁，不受红色区域阻挡的路径就越常出现。这项实验证实了我们的假设：自适应智能体和非自适应智能体不会发生任何状态违规。

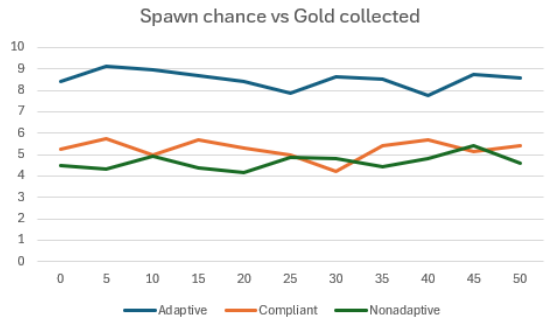
总体而言，与其他智能体相比，自适应智能体平均实现的目标更多，同时不会发生状态违规，且产生的罚分约为遵从型智能体的一半。非自适应智能体产生的罚分最少，但实现的目标也少于其他智能体。

6.5 实证结果——怪物

实验结果： 图5a显示了每个智能体实现的目标平均数量。自适应智能体（即 Coward）取得了最高平均值；Compliant 实现的目标略多于非自适应智能体。该实验验证了我们的假设：Adaptive 实现的目标多于 Nonadaptive 和 Compliant。同样，如图5b所示，自适应智能体收集的金币多于 Compliant 和非自适应智能体，而后两者在这一指标上的表现相近。Nonadaptive 与 Adaptive 在两个指标——实现的目标和收集的金币——上存在差异的原因在于，非自适应智能体一遇到任何怪物就会停止。相比之下，除非怪物位于金子位置上方，否则自适应智能体会设法绕过怪物。Compliant 与 Adaptive 之间的差异，是由于 Compliant 更频繁地死亡：它会在前往指定金子位置的路径上与任何怪物交战。图6a对此进行了说明，其中显示 Compliant 在超过一半的回合中死亡。图6b也对此进行了说明，其中显示 Compliant 的偏差接近每回合一个 $\mathcal{D}$-偏差



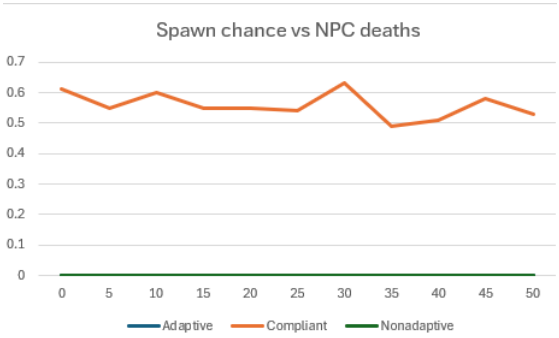
（a）横轴表示怪物生成的概率（每个刻度一次）。纵轴表示实现的目标数量，即每种智能体类型经过100个回合后的平均值。



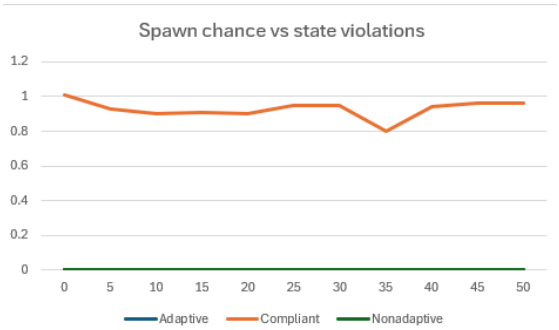
(b) 横轴表示每个刻度上怪物生成的概率。纵轴表示每种智能体类型在100个回合中平均收集的黄金总量。

图5：性能指标随怪物生成概率的变化。

回合。相比之下，Adaptive和Nonadaptive从未出现 $\mathcal{D}$差异，从而证实了我们的实验假设：这两种智能体不会导致状态违规。



(a) 横轴表示每个刻度上怪物生成的概率。纵轴表示每种智能体类型在100个回合中的平均智能体死亡数。



(b) 横轴表示每个刻度上怪物生成的概率。纵轴表示每种智能体类型在100个回合中的平均状态违规数。

图6：性能指标随怪物生成概率变化的情况。

总体而言，我们从各项指标中观察到，智能体的表现与其特征一致。Adaptive（Coward）智能体从不死亡，并收集到最多的金币（尽管它并不总是达成其指定目标）。Nonadaptive 智能体同样从不死亡，但由于遇到任何怪物都会导致该智能体放弃达成其目标，因此它收集到的金币较少。最后，Compliant 智能体将“盲目地”追求其指定的金币，从而更频繁地死亡，结果收集到的金币少于 Adaptive 智能体。

7. 相关工作

我们的反叛模型可以与人类认知理论联系起来。在《人因》研究中，明智不服从（Chaleff，2015）描述了操作员或助手如何接受训练，以便在命令与安全规则相冲突时推翻这些命令。一个众所周知的例子是导盲犬，它们被训练为：如果服从主人的命令会使主人陷入危险，就拒绝该命令。例如，狗会在繁忙的十字路口停下并拒绝继续前进。同样，在军事领域，一名有纪律的士兵如果服从命令会导致危害人类罪，就应当拒绝命令。在我们的工作中，我们为智能体赋予指令 D，使得当用户的命令会导致违规时，智能体能够进行调整，以避免造成违规，同时仍力求实现用户的意图。这类似于狗等到过马路安全时才行动。Chaleff（2015）将这种行为称为“暂停并评估”，强调明智不服从涉及审议，而非盲目服从。

认知控制的心理学模型（例如，执行功能中的冲突监测）解释了人类如何检测目标与约束之间的矛盾，并开展替代规划（Greene et al., 2001）。一个说明性案例是载有一名乘客的自动驾驶汽车所面临的道德困境；它必须在偏离路线、可能伤害其乘客，或继续行驶并可能在交叉路口伤害多名行人之间做出选择。在我们的语境中，这对应于 D 中有两条或更多指令被同时触发的情形，要求智能体在相互竞争的规范或伦理约束之间确定优先级。在我们工作的当前版本中，智能体通过任意选择一条指令来解决此类冲突（第28行和第31行）。在未来工作中，我们将探索以更有原则的方式解决指令冲突的机制，并从人类道德推理模型中汲取灵感。

术语反叛智能体 由Coman与Muñoz-Avila（2014）提出。该工作的动机来自数字游戏，以及非玩家角色在接收玩家指令时需要以更合理的方式行事。它采用了戏剧管理器的思想（Yu & Riedl, 2013），用以评估智能体的动机是否与用户的指令相冲突。例如，用户可能命令智能体通过偷窃来获取一件物品，但这与智能体的“守序善良”阵营相矛盾（Gygax（1978）；第23页）。Coman与Muñoz-Avila（2014）还为反叛智能体奠定了基础，即智能体将规划结果与自身动机进行核对。在本文中，我们转而在在线分层任务网络规划的语境下形式化了 $\mathcal{D}$偏差的思想。在先前工作中，研究者在目标驱动自主（Molineaux et al., 2010）的目标推理模型语境下定义了反叛智能体。因此，当发生 $\mathcal{D}$偏差时，那些模型会生成一个新目标。相比之下，我们的自适应智能体仍会尝试通过以不与 $\mathcal{D}$ 相矛盾的方式调整计划来实现目标（即，它们将尝试重新规划，而不是放弃该目标）。最近，人们对反叛智能体这一主题的兴趣有所增加。更广泛的讨论见（Coman & Aha, 2018）。

我们的工作也与多智能体系统（MAS）中的规范有关（Sergot，2007）。规范是智能体在环境中执行动作时应遵守的原则。已有大量研究探讨多智能体系统中的规范（Savarimuthu 和 Cranefield，2011）。我们的工作并未规定如何生成 $\mathcal{D}$。因此，可以设想 $\mathcal{D}$ 会涵盖特定领域的规范。我们唯一的要求是，可以对 $\mathcal{D}$ 进行测试，以确定其在某个状态下是否遭到违反。这样，智能体就可以针对当前状态或投影状态进行测试。

我们的工作也与 BDI（信念-愿望-意图）模型有关（Georgeff等人，1999）。这些模型旨在监控并确保计划的执行能够实现其意图，并在这些意图未实现时标注替代方案。在我们的工作中，我们使用分层任务网络进行计划生成；由此产生的任务层级和 $\mathcal{D}$ 在计划生成过程中发挥着类似的作用。例如，动作 *down(3)* 可以解释为智能体正试图完成任务 *reach(3,pink)*。如果出现 $\mathcal{D}$-偏差，则可以使用分层任务网络重新规划来修订计划。

我们的在线HTN规划方法符合行动者视角规划范式（Ghallab等人，2014）。该范式提倡采用HTN规划，将规划与执行交错进行；这尤其适用于本文所研究的这类环境，因为不断变化的状态动态可能要求智能体在计划执行期间重新规划。例如，在Yuan等人（2022）的研究中，HTN计划生成与计划执行交错进行；当环境中的动态变化导致动作变得不适用时，便会修改任务。但该研究并未考虑指令，因此任务修改是由环境条件变化导致的，而不是因为违反了 $\mathcal{D}$。另一个区别是，该研究没有进行前瞻，以考虑未来状态中可能出现的违规。

8. 结论

我们介绍了叛逆在线HTN规划智能体，这类智能体能够对指令 $\mathcal{D}$进行推理。指令使这些智能体能够评估是否会出现状态差异，并在必要时重新规划。我们说明了HTN规划智能体如何利用 $\mathcal{D}$ 防止安全领域中出现此类差异，以及如何利用 $\mathcal{D}$ 赋予非玩家角色人格特质。通过在O-RESCHU任务领域开展实验，我们确认这些智能体从不违反 $\mathcal{D}$。此外，我们还研究了两种变体：Nonadaptive和Adaptive；后者会在必要时动态修改其任务列表，以生成不违反 $\mathcal{D}$的备选计划。我们发现，与遵从型智能体相比，Adaptive智能体以更低的成本（即罚分数）实现了大部分分配的目标。Nonadaptive智能体的成本最低，但实现的目标数量也最少。在MONSTER领域，我们观察到了类似的结果：自适应智能体收集到更多金币且从未死亡；Nonadaptive智能体收集的金币较少，但同样从未死亡；遵从型智能体则频繁死亡，收集的金币也较少。

潜在的未来研究方向包括研究在执行更多动作之前进行前瞻的智能体。这对于智能体可能发现自己无法回溯的领域是必要的。例如，在 RESCHU 领域的一个变体中，智能体可能不被允许沿原路返回，而且红色区域可能在彼此重叠的区域生成。在这种情况下，智能体需要向前生成n 步，π_n（其中 $n \geq 2$），以检查可能的 $\mathcal{D}$差异，而不执行这些动作。另一个潜在的未来研究方向是考虑具有数值流的 $\mathcal{D}$（例如汽油消耗），以及能够明确推理相关 $\mathcal{D}$差异的智能体。我们还将研究此情境下的非确定性，即动作可能有多个结果。例如，由于风况的影响，当智能体从单元格 $(x,\ y)$ 的某个位置执行“向上”动作时，可能会发现自己位于位置 $(x,\ y-1)$ 或 $(x+1,\ y-1)$。我们认为，如果某个动作的潜在结果会导致 $\mathcal{D}$ 违规（例如，格子 $(x,\ y)$ 或 $(x+1,\ y-1)$

位于红色区域内，那么智能体就会考虑备选动作）。但是，如果智能体需要对概率性结果进行推理（例如，认为不违反 $\mathcal{D}$ 的概率为 95% 时可以接受），则还需要进一步研究。在这种情况下，智能体必须推理到达发生违规状态的概率。其他未来研究工作还包括分析在线分层任务网络规划相较于线性重新规划所带来的计算收益，并通过比较不同求解器的重新规划频率和所需计算时间进行评估。

另一个值得研究的指标是，在不同类型的约束下生成反叛行为所需的审议量。例如，由于能量点损失而“避开红色区域”的指令可以作为一种安全约束被编译消除，除约束满足之外不需要额外的审议。然而，如果无人机断定自己必须穿越红色区域以避免坠毁，就会发生反叛行为。自适应智能体是在智能反叛/不服从语境中对这类情形进行推理的第一步，未来工作可以处理各种形式的约束，例如竞争目标、不同利益相关者提供的指令以及伦理禁令。

在我们当前的框架中，我们对指令的内部表示不作任何假设。我们只要求每条指令 $\delta \in D$ 都能作为给定状态 s上的布尔函数进行求值，并返回该指令是被满足还是被违反。从这个意义上说，指令被视为我们的R-HTN智能体的输入，而不是智能体进行推理的对象。未来工作的另一条途径是探索更具表达力的指令表示。例如，基于逻辑的形式体系或基于规范的表示可以使智能体推断更高层的属性，如伦理约束或规范性优先级，并审议相互竞争的指令之间的权衡。这类方法还将使对指令层级与冲突的研究成为可能，从而使我们的工作更直接地与道德推理理论和规范性决策联系起来。

最后，Mirsky等人（2025）提出了设计不服从用户的智能体时应具备的三个关键要素，这些要素同样适用于我们的反叛智能体。首先，智能体应当透明，说明其不服从命令的原因。其次，智能体应当合理可辩，也就是说，不服从的理由应当清晰，并以站得住脚的原则为依据。第三，智能体应当可控，允许用户在需要时推翻智能体的不服从行为。未来一个值得探索的方向是让反叛智能体具备这三个要素，从而提升其在人机交互中的易用性和可信度。

参考文献

Boussemart, Y., & Cummings, M.（2008）。操作员监督多种异构无人车辆的行为识别与预测。操作无人系统的人类。Cardoso, R. C., & Bordini, R. H.（2017）。一种分层任务网络规划形式体系的多智能体扩展。*ADCAIJ*：分布式计算与人工智能进展期刊。Chaleff, I.（2015）。明智不服从：当你被告知去做的事是错的时，做正确的事。加利福尼亚州奥克兰：Berrett-Koehler 出版社。Coman, A., & Aha, D. W.（2018）。AI反叛智能体。《人工智能杂志》，*39*，16–26。

Coman, A., & Muñoz-Avila, H.（2014）。反叛智能体的动机差异：构建基于案例的目标驱动自主性框架以提升角色可信度。第*22*届国际案例推理会议（*ICCBR*）基于案例的智能体研讨会论文集。Donaldson, M.（2014）。使用层次任务网络对*combatxxi*中的动态战术行为进行建模。硕士论文，美国海军研究生院。Gartenberg, D., Breslow, L., McCurry, J. M., & Trafton, J. G.（2014）。情境意识恢复。《人因》，*56*，710–727。Georgeff, M., Pell, B., Pollack, M., Tambe, M., & Wooldridge, M.（1999）。能动性的信念—愿望—意图模型。《智能体 *V*：智能体理论、架构与语言：第*5*届国际研讨会 *ATAL'98*，法国巴黎，*1998*年*7*月*4*日至*7*日，论文集 *5* （第1–10页）。Springer。Georgievski, I., & Aiello, M.（2015）。分层任务网络规划：概述、比较与展望。《人工智能》，*222*，124–156。Ghallab, M., Nau, D., & Traverso, P.（2014）。自动规划与行动的行动者视角：一篇立场论文。《人工智能》，*208*，1–17。Greene, J. D., Sommerville, R. B., Nystrom, L. E., Darley, J. M., & Cohen, J. D.（2001）。道德判断中情绪投入的功能性磁共振成像研究。《科学》，*293*，2105–2108。Gygax, G.（1978）。《高级龙与地下城》。TSR Hobbies。Huang, X.（2025）。小型无人机革命即将到来——科学家需要确保它的安全。《自然》，*637*，29–30。Mirsky, R., Shapiro, D., & Rahwan, I.（2025）。人工智能抗命：重新思考人工智能队友的能动性。《人工智能杂志》，*46*，19–30。Molineaux, M., Klenk, M., & Aha, D.（2010）。海军战略模拟中的目标驱动自主。*AAAI*人工智能会议论文集（第1548–1554页）。Musliner, D. J., & Goldman, R. P.（2010）。分层任务网络规划中的基于优先级的元控制。*2010*年第四届*IEEE*自适应与自组织系统国际会议研讨会（第282–286页）。IEEE。Nau, D., Cao, Y., Lotem, A., & Munoz-Avila, H.（1999）。SHOP：简单层次有序规划器。第*16*届人工智能国际联合会议论文集——第*2*卷 （第968–973页）。Savarimuthu, B. T. R., & Cranefield, S.（2011）。规范创建、传播与涌现：多智能体系统中规范仿真模型综述。《多智能体与网格系统》，*7*，21–54。Sergot, M.（2007）。规范治理的多智能体系统中的动作与能动性。智能体世界中的社会工程国际研讨会 （第1–54页）。Springer。Smith, S. J., Nau, D. S., Throop, T. A., 等（1998）。黑桃游戏中的成功：运用人工智能规划技术赢得世界计算机桥牌锦标赛。*AAAI/IAAI*，*98*，1079–1086。

Verweij, T.（2007）。面向第一人称射击游戏的分层多人机器人系统。硕士论文，阿姆斯特丹自由大学。Yu, H. 与 Riedl, M. O.（2013）。数据驱动的个性化戏剧管理。第九届*AAAI*人工智能与互动数字娱乐会议论文集。Yuan, W.、Munoz-Avila, H.、Gogineni, V. R.、Kondrakunta, S.、Cox, M. 与 He, L.（2022）。用于分层任务网络规划与行动的任务修饰符。第九届认知系统（*ACS*）进展会议。