

基于多策略 Q-learning 算法的连续动作优化模型

摘要

游戏是智能学习的重要表现之一，本文建立了多策略 Q-learning 算法的连续动作优化模型，采用动态 ϵ -greedy 策略、期望学习策略与博弈策略求解玩家穿越最优路径。

针对问题一，建立固定环境参数的连续动作优化模型，并使用 Q-learning 算法对该模型进行优化求解。首先将附件中的地图数据转换为代自环的连通图矩阵，将问题中的背包容量、天气因素与生存条件等题目信息刻画为约束条件，并采用精确购买策略。再在连通图矩阵中添加时间维度使得矩阵转换为三维 Q 矩阵，并采用动态 ϵ -greedy 策略探索更新 Q 矩阵直至其收敛。最后通过完全贪婪策略输出 Q 矩阵表示的最优动作组合，并求出玩家每日物资剩余量。其第一关最优策略为 [1, 25, 24, 23, 23, 22, 9, 9, 15, 15, 15, 15, 13, 12, 12, 12, 12, 12, 12, 12, 12, 13, 15, 9, 9, 21, 27]，所得的最大保留资金为 10590 元；第二关最优策略为 [1, 9, 10, 19, 19, 27, 36, 36, 44, 53, 54, 54, 62, 55, 55, 55, 55, 55, 55, 55, 55, 62, 55, 55, 55, 55, 55, 56, 64]，所得最大保留资金为 12460 元。

针对问题二，本问在问题一模型的基础上对天气变量随机化，并通过加入期望学习策略的改进 Q-learning 算法使其适应随机环境参数。在三维 Q 矩阵的基础上添加天气变量维度，并使用最优 Q 值的期望作为经验学习项更新 Q 矩阵。对于第三关，不论天气情况如何更变，最优路径始终为 [1, 4, 6, 13] 与 [1, 5, 6, 13]。对于第四关，玩家通关概率与平均收获将随高温、沙暴概率增大呈现下降趋势。

针对问题三第五关，使玩家间构成完全竞争关系，建立静态的完全信息变和博弈模型。在命题证明的基础上求解玩家可采用的最优策略仅有 [1, 4, 6, 13] 与 [1, 5, 6, 13]，当两个玩家选择相同策略，各自保留资金均为 7640 元，当两个玩家选取不同最优路径时，各自保留资金均为 8840 元；针对问题三第六关，建立三人合作模型，基于 Q-learning 算法并使用谦让策略选择最优路径。仿真结果显示表示执行谦让策略后，即使天气情况较差时部分玩家也可保持较高通关率。

本文的优点为：1. 基于期望学习策略的 Q-learning 算法可适应带有随机变量的环境参数，并通过奖励期望值对 Q 矩阵进行更新。2. 改进 Q-learning 算法的时间复杂度为 $O(n)$ 远优于一般群集智能优化算法，并且每个 agent 可通过不同的策略自行探索动态学习，兼顾局部搜索与全局搜索能力。

关键词：改进 Q-learning 动态 ϵ -greedy 策略 完全信息变和博弈 期望学习策略

1 问题重述

1.1 问题背景

在“穿越沙漠”游戏中，玩家每天可以移动至相邻区域或停留在原地，并根据天气选择是否继续前行，在满足背包容量的前提下携带水和食物两种生活物资。玩家到达矿山后，可以选择挖矿获取资金收益；抵达村庄时可补充物资。按照给定地图，在一定的时间约束内抵达终点并保留最多资金的玩家获胜。

游戏中，需要分别综合考虑物资消耗与背包容量、挖矿消耗与资金收益、天气变化与行程决策等目标与约束的关系。同时，不同关卡中具有不同的给定地图与截止日期，需要在满足背包容量约束、时间约束等约束的条件下，使保留资金尽可能多。

本文针对不同的关卡，考察单玩家和多玩家的游戏类型，分别讨论事先是否知晓天气状况下的玩家策略，得到不同背景下“穿越沙漠”游戏最佳取胜策略。



图 1 问题背景描述图

1.2 问题概述

围绕相关附件和条件要求，研究不同背景下“穿越沙漠”游戏最佳取胜策略，依次提出以下问题：

问题一：在已知整个游戏时段的天气状况的情况下，给出单玩家最优游戏策略。

问题二：在仅知当天天气状况的情况下，给出单玩家最佳游戏策略。

问题三：考虑 n 名玩家共同游戏，共同挖矿或在同一村庄购买生存物资，则基础资金降低或花费成本按规律增大。在不同的地图中，分别考虑已知整个游戏时段的天气状况和仅知当天天气状况的情况，给出相应的最优游戏策略。

2 模型假设

- (1) 假设负重情况不会影响玩家的体力消耗，即无论负重量为多少，玩家的行动能力与食物、饮水的消耗量都不会发生改变。
- (2) 当天气情况未知时，假设玩家能知晓接下来各种天气发生的大致概率。
- (3) 假设在第五关时，玩家之间构成全面竞争关系，即胜利条件为最终保留资金大于另一个玩家。
- (4) 假设在第六关时，玩家为了保证通关，将默认与其他玩家进行合作，即对其他玩家建模并执行谦让策略。

3 符号说明

符号	说明
s_t, a_t	第 t 天时出发地点与采取的行动
$weather_t$	第 t 天的天气情况
$consume(weather_t)$	基础水与食物所对应资金消耗
$\Delta w(weather_t), \Delta f(weather_t)$	基础饮水与食物消耗
\otimes	表示执行动作
M, R	惩罚函数，奖励函数
$P(weather_t)$	第 t 天天气概率函数

注：表中未说明的符号以首次出现处为准

4 问题一模型的建立与求解

4.1 问题描述与分析

问题一要求在天气情况事先全部已知的情况下，分别给出玩家在第一关与第二关中的最优策略。鉴于游戏目标为在规定时间内到达终点并保留尽可能多的资金，即将优化目标定义为到达终点时玩家的剩余资金，同时必须满足食物与水资源充足等生存约束条件，即可得到沙漠穿行策略优化模型。由于天气情况事先全部已知，玩家可根据穿行计划精确购买需求物资，即可将每天行动的物资消耗等价转化为资金消耗。

不难得出，此时玩家的资金变化仅源于物资消耗与挖矿收益，基于此，本文采用 Q-learning 算法优化求解最优策略。在 Q-learning 算法中，首先以算法中的 agent 表示玩家，将其各天的所在位置与采取的行动分别定义为状态变量与动作变量，将各状态采取行动后的净收益定义为奖励函数，并采用动态 $\varepsilon - greedy$ 原则迭代优化 Q 矩阵直至其收敛。

其思维框图如图 2 所示：

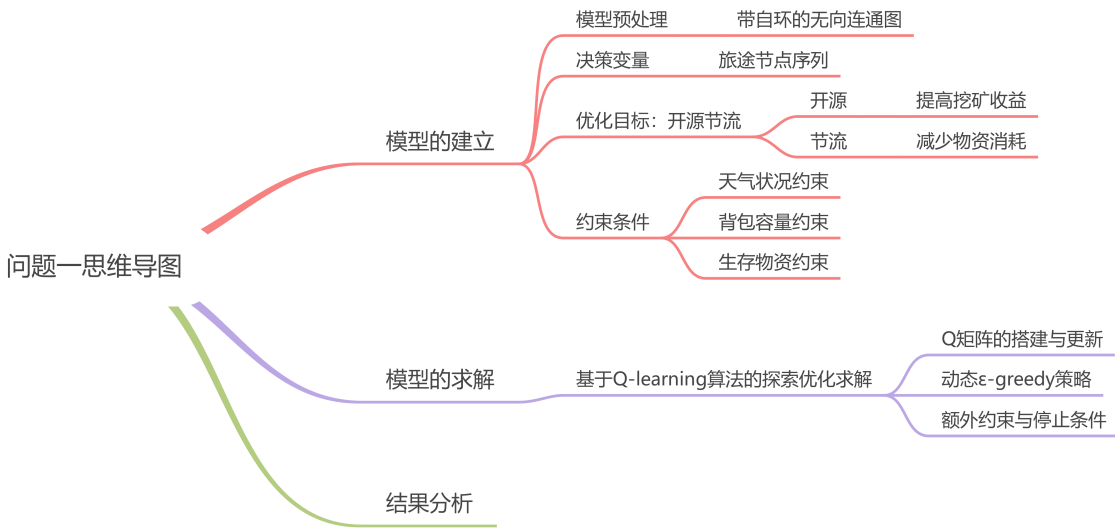


图 2 问题一思维流程图

4.2 固定环境参数的连续动作优化模型的建立

4.2.1 模型预处理

所有关卡地图可以表示为带自环的无向连通图，记为 $G(V, E, \mathbf{W})$ 。设连通图阶数为 n ，则每个区域为节点 $v_i \in V (i = 1, 2, \dots, n)$ ，相邻区域间通道可以表示为边 $e_j \in E (j = 1, 2, \dots, m)$ ，权重矩阵为 \mathbf{W} ，存放玩家经过相邻通道或停留在原先区域的消耗量代价。连通图可表示如图 3 (b) 所示

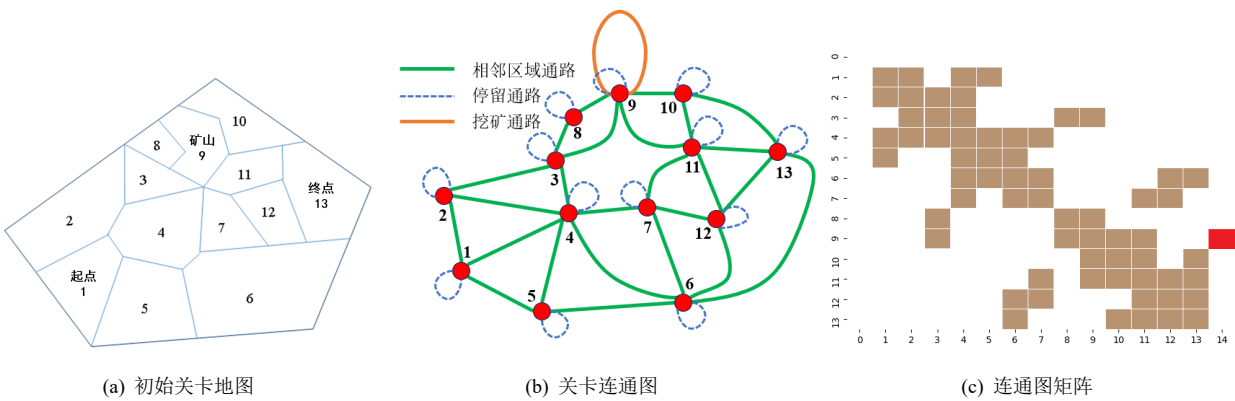


图 3 问题一图论模型示意图

如图，绿色边表示两地点间存在的可行路径。为表示玩家停留于原先区域的行为，使每个节点带自环，即图中的蓝色虚线边，且自环边也赋有对应消耗权重。玩家在矿山停留时，为区别休息与挖矿的行为，在连通图中矿山节点上加入两种自环边分别表示挖

矿与休息行为，即当玩家进入矿山时，选择挖矿行为即选择橙色自环，选择休息即选择蓝色自环。

之后即可将联通图转换为连通图矩阵如图 3 (c) 所示，其中深色块表示对应两点相连，无色色块表示对应两点没有直接相连，且矿山节点后存在一额外色块表示挖矿动作代表的自环通路。

4.2.2 决策变量

设玩家经过 n 天旅行到达终点，则行程路线由玩家经过的节点序列构成，路线的每个元素为途经的节点序号 $s_t (t = 1, 2, \dots, n)$ ，采用动作为 $a_t (t = 1, 2, \dots, n)$ 。记截止时间为 T ，则决策变量可以表示为

$$(S_{n+1}|A_n) = \begin{bmatrix} s_1 & s_2 & \dots & s_n & s_{n+1} | a_1 & a_2 & \dots & a_n \end{bmatrix}, \quad (1)$$

$$1 \leq n \leq T, \quad (2)$$

其中各个 s_t 表示玩家第 t 天时玩家的出发地点，也即是第 $t - 1$ 天的停留地点， a_t 表示第 t 天玩家所采取的行动，包括行动、停留以及挖矿，其满足

$$s_{t+1} = s_t \otimes a_t.$$

定义符号 \otimes 表示执行动作，即 s_t 在执行动作 a_t 后即转化为状态 s_{t+1} 。特别的， s_1 与 s_{n+1} 表示游戏开始时玩家在初始起点位置与第 n 天时玩家须到达终点位置，即须满足

$$\begin{cases} s_1 = s_{outset}, \\ s_{n+1} = s_{destination}. \end{cases}$$

4.2.3 约束条件

问题中，由于生存物资的消耗与补给与行程密切相关，而行程路线由天气情况决定，同时天气情况影响生存物资的消耗，故需要整理约束条件的分类，综合考虑几大约束的关系，并从多个维度考虑各约束条件对行程决策的影响。

1. 行动受天气约束的刻画

定义第 t 天的天气变量为 $weather_t = \{1, 2, 3\}$ ，即沙暴、高温与晴朗时的天气变量 $weather_t$ 的值分别为 1、2 与 3。此时动作变量需满足约束如下

$$|a_t| \leq \frac{weather_t}{2},$$

其中 $|a_t|$ 表示动作 a_t 的移动量，其仅可取值为 0 和 1 即表示停留原地和移动。

2. 背包容量约束的刻画

通过函数表示旅途中背包中剩余物资量，进而推导背包约束关系。记第 t 天背包中剩余水和食物量分别为 $w(t)$ 和 $f(t)$ ，两者是离散化的关于时间 t 的函数。旅程中的任何时刻，水和食物两种生存物资的装载量需小于背包容量 b_0 ，即有

$$\forall t = 0, 1, 2, \dots, T : w(t) + f(t) \leq b_0.$$

3. 生存物资约束的刻画

游戏过程中，如果未到达终点而水或者食物已经耗尽，视为游戏失败。故需要满足水和食物的剩余量在游戏过程中始终非负，即

$$\forall t = 0, 1, 2, \dots, T : \begin{cases} w(t) \geq 0, \\ f(t) \geq 0. \end{cases}$$

第 t 天的饮水于物资消耗可表示为

$$\begin{cases} w(t+1) = w(t) - \Delta w(weather_t) \cdot \lambda(s_t, a_t), \\ f(t+1) = f(t) - \Delta f(weather_t) \cdot \lambda(s_t, a_t). \end{cases} \quad (3)$$

$\Delta w(weather_t)$ 与 $\Delta f(weather_t)$ 分别表示在第 t 天天气为 $weather_t$ 情况下时的基础饮水与食物消耗。 $\lambda(s_t, a_t)$ 是关于状态变量 s_t 、动作变量 a_t 的消耗倍率函数，即

$$\lambda(s_t, a_t) = \begin{cases} 3, & s_t = s_{mine} \wedge a_t = a_{mining}, \\ 2, & a_t = a_{travel}, \\ 1, & otherwise. \end{cases}$$

即当玩家在矿山挖矿时，消耗为基础消耗的三倍；当其选择移动时，消耗为基础消耗的两倍；当其选择在原地停留时，消耗等于基础消耗。

4.2.4 目标函数

当玩家在状态 s_t 下执行动作 a_t 后即可得到当天的净收入为

$$R(s_t, a_t, weather_t) = -consume(weather_t) \cdot \lambda(s_t, a_t) + income(s_t, a_t), \quad (4)$$

其中 $consume(weather_t)$ 表示在第 t 天天气为 $weather_t$ 情况下时的基础饮水与食物消耗所对应的资金消耗， $\lambda(s_t, a_t)$ 是关于状态变量 s_t 、动作变量 a_t 的消耗倍率函数。

$income(s_t, a_t)$ 为挖矿收益函数，可表示如下

$$income(s_t, a_t) = \begin{cases} \tau, & s_t = s_{mine} \wedge a_t = a_{mining}, \\ 0, & \text{其他.} \end{cases}$$

其中 τ 为一次挖矿的收益，即仅当玩家在矿山挖矿，且执行挖矿操作时，其才可获得挖矿收益，否则收益为零，由于第 t 天的天气情况提前已知即天气信息包含于状态信息内 $weather_t \in s_t$ ，即 $R(s_t, a_t, weather_t)$ 可简化表示为 $R(s_t, a_t)$ 。玩家的总体目标函数，即到达终点时的剩余资金可表示为

$$\max_{(S_{n+1}|A_n)} M_0 + \sum_{t=1}^n R(s_t, a_t), \quad (5)$$

其中 M_0 表示玩家的初始资金， $\sum_{t=1}^n R(s_t, a_t)$ 表示从第 1 至 n 。

4.2.5 固定环境参数的连续动作优化模型

以游戏过程中保留资金最多为目标，结合各约束条件，给出最佳游戏策略，整体优化模型可表示为

$$\begin{aligned} & \max_{(S_{n+1}|A_n)} M_0 + \sum_{t=1}^n R(s_t, a_t), \quad (6) \\ s.t. & \begin{cases} 1 \leq n \leq T, \\ s_{t+1} = s_t \otimes a_t, \\ s_1 = s_{outset}, \\ s_{n+1} = s_{destination}, \\ |a_t| \leq \frac{weather_t}{2}, \\ \forall t = 0, 1, 2, \dots, T : w(t) \geq 0, \\ \forall t = 0, 1, 2, \dots, T : f(t) \geq 0, \\ \forall t = 0, 1, 2, \dots, T : w(t) + f(t) \leq b_0, \\ w(t+1) = w(t) - \Delta w(weather_t) \cdot \lambda(s_t, a_t), \\ f(t+1) = f(t) - \Delta f(weather_t) \cdot \lambda(s_t, a_t), \\ R(s_t, a_t, weather_t) = -consume(weather_t) \cdot \lambda(s_t, a_t) + income(s_t, a_t). \end{cases} \quad (7) \end{aligned}$$

4.3 基于 Q-learning 算法的模型求解

针对连续动作模型的优化求解，本文以算法中的 agent 代表玩家，采用 Q-learning 算法^[1] 对目标模型进行探索优化求解具体步骤可表示如下

step1: Q 矩阵搭建

即首先建立 Q 矩阵，鉴于状态变量 s_t 所包含信息有当前时间与当前位置两种信息，我们将建立三维矩阵如下图 4 所示

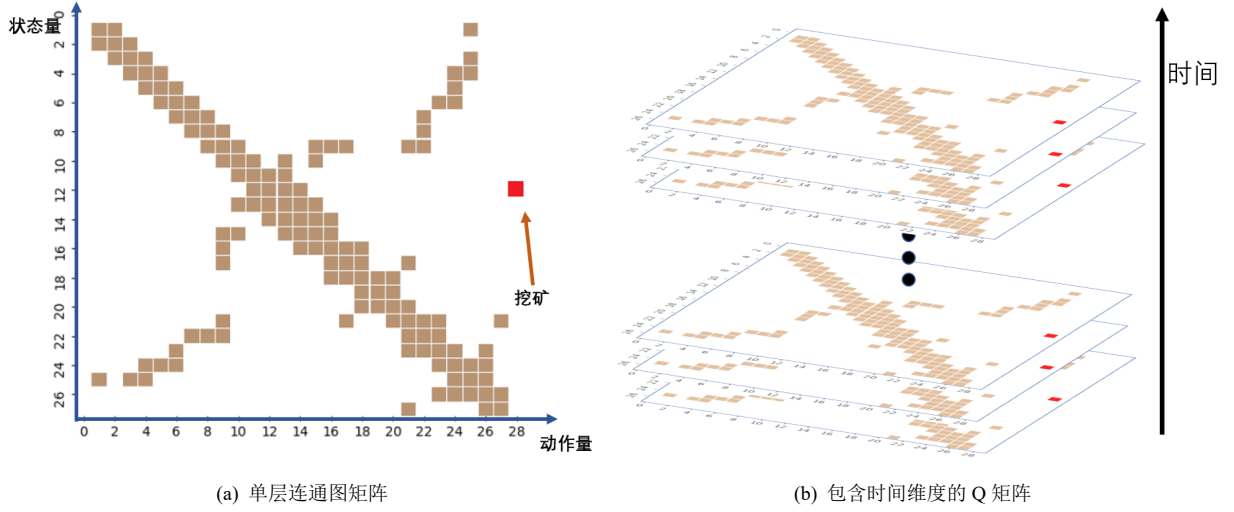


图 4 三维 Q 矩阵示意图

图 4 (a) 中矩阵为沙漠地图所表示的可自环连通图转化成的连通图矩阵，其表示图 4 (b) 中高维 Q 矩阵中的某一层，有色色块处填充初始值为 0，无色色块处填充初始值为 $-\infty$ 。设置算法的最大探索次数为 ξ ，当前探索次数为 k ，即当 $k \leq \xi$ 时，进入迭代探索，初始化 agent 的位置坐标，即令 $s_0 = s_{outset}$ 。

step2: 动态 ϵ -greedy 策略

即算法在第 k 次迭代中，将以 $\epsilon(k)$ 的概率执行随机探索策略，即在状态变量为 s_t 时等概率的随机选取动作 a_t 作为执行的动作变量，并以 $1 - \epsilon(k)$ 的概率执行贪婪策略，即选取对应 Q 值最大的 a_t 作为动作变量。特别的，即不论 a_t 取何值， s_t 中的时间维度 t 必将增加 1，即 $t \leftarrow t + 1$ ，即状态变量将移动至第 $t + 1$ 层矩阵中。并定义 $\epsilon(k)$ 函数如下所示

$$\epsilon(k) = \begin{cases} 1 - \frac{k}{T_0}, & k < T_0, \\ 0, & k \geq T_0. \end{cases} \quad (8)$$

其中 T_0 表示执行动态 ϵ -greedy 策略的迭代次数，当 $k \geq T_0$ 时策略转化为完全贪婪策略。

step3: Q 矩阵的更新

当策略为最优策略 π^* 时， s_t 处的价值函数 $V^*(s_t)$ 可通过 Bellman 优化方程^[1, 3] 表示为

$$V^*(s_t) = \max_{a_t \in A(s_t)} \mathbb{E}_{S_{t+1} \sim E}[R(s_t) + \gamma \max V^*_{s_{t+1}}], \quad (9)$$

其中 $\gamma \in [0, 1]$ 表示折扣因子，其值越大 agent 就将越重视记忆中的最优策略，其值越小，agent 将越重视眼前利益。此时，最优的 Q 函数可被表示为

$$Q^*(s_t) = \mathbb{E}_{s_{t'} \sim E}[R(s_t) + \gamma \max V^*_{s_{t'}}], \quad (10)$$

其中 t' 表示第 n 天内的任意时刻，此时 Q 函数的更新方式可转换如下所示

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \lambda[R(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)], \quad (11)$$

$\lambda \in [0, 1]$ 表示学习速率，其表示 Q 矩阵的更新速度。即最优策略可被表示为

$$\pi^*(s_t) = \arg \max_{a_t \in A(s_t)} Q(s_t, a_t).$$

step4: 额外约束与探索终止条件

当 $s_t = s_{destination}$ 时，即玩家在第 t 天到达终点时即停止第 k 次探索，此时有

$$\max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) = 0,$$

即由于到达终点时第 k 次探索终止，此时不存在下一动作 a_{t+1} ，记忆中的最优动作 Q 值为 0。若玩家处于状态 s_t 并执行动作 a_t 后，存在 $w(t) < 0$ 或 $f(t) < 0$ 时，即食物或饮水资源耗尽时停止迭代，并在奖励函数 $R(s_t, a_t)$ 上添加惩罚项如下

$$R(s_t, a_t) = R(s_t, a_t) + \min\{0, f(t)\} M_1 + \min\{0, w(t)\} M_2, \quad (12)$$

其中 M_1 与 M_2 为较大的正数，即可起到惩罚项的作用。当 $t = T$ 时，若执行完动作变量 a_t 后的状态变量 $s_{t+1} \neq s_{destination}$ 则停止迭代且在奖励函数 $R(s_t, a_t)$ 上添加惩罚项如下

$$R(s_t, a_t) = R(s_t, a_t) + \theta,$$

其中 θ 为绝对值较大的负数，即作为超过比赛截至日期的惩罚项。

则动态 ϵ -greedy 策略的算法流程图如 5 所示

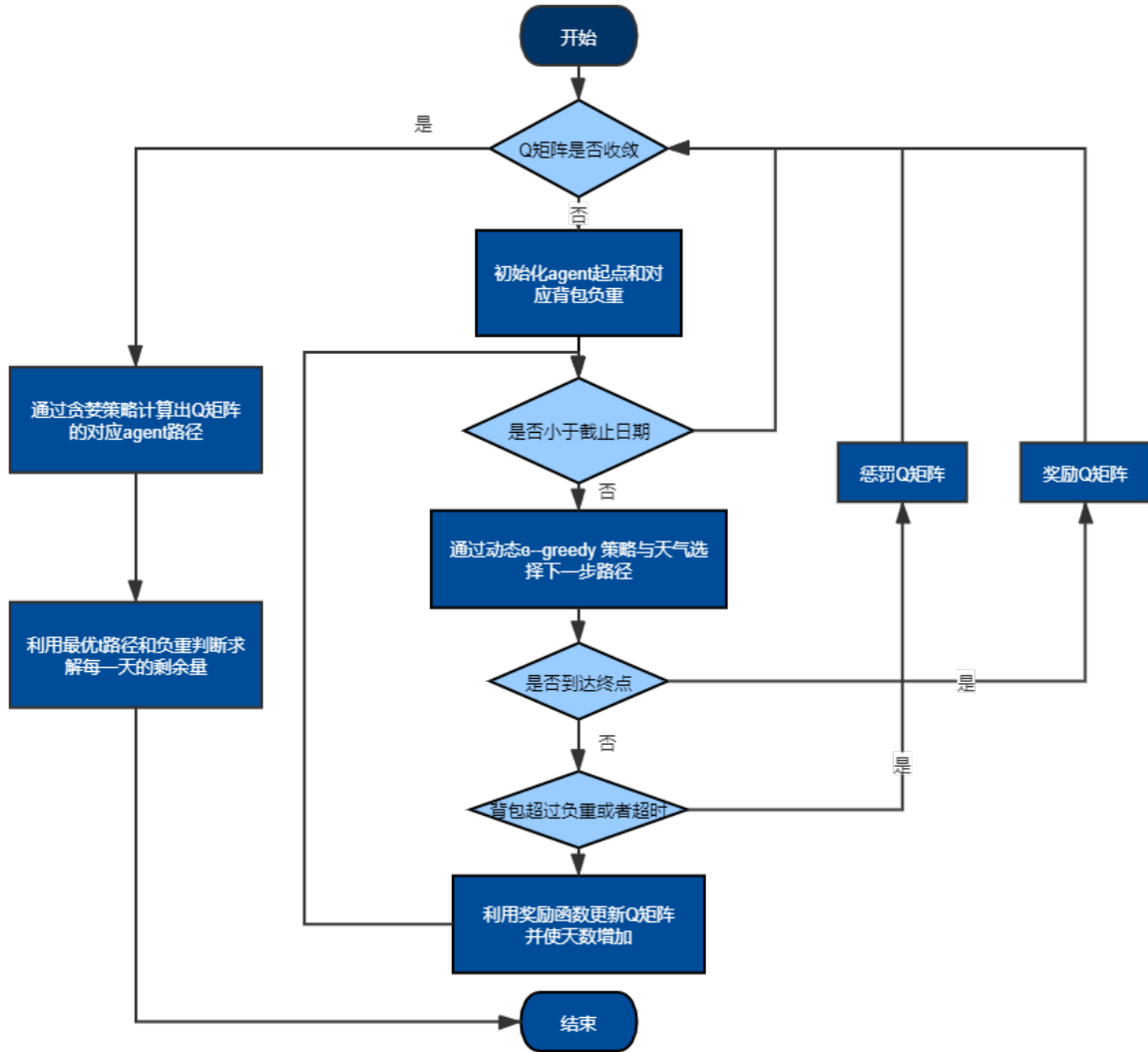


图 5 Q-learning 求解连续动作优化模型流程图

4.4 实验结果及分析

由于在关卡一中其地图中矿山距离起点与终点位置较远，通过动态 ϵ -greedy 策略执行的 Q-learning 算法容易陷入到局部最优解（其路径为 [1,25,26,27]）。为跳出该局部最优解，本文首先通过计算最小路径算法使起始位置于村庄（其路径为 [1, 25, 24, 23, 23, 22, 9, 9, 15]），再执行 Q-learning 算法，从而减少问题规模并降低算法中的无效探索比率。设置初始参数 $\lambda = 0.8, \gamma = 0.8$ ，迭代次数 $k=100000$ ，设置执行动态 ϵ -greedy 策略的迭代次数为 $T_0 = 0.9k$ 。

由于动态 ϵ -greedy 策略的探索过程具有不确定性，每次运行算法收敛到的 Q 矩阵具有不唯一性。反复执行算法后求解得出第一关的最优结果如表 2 所示，该路径求出最大收益资金为 590 元，其剩余量见附件 Result.xlsx。

表 1 第一关该玩家穿越路径最优策略

天数	天气	行为状态	节点移动状态	天数	天气	行为状态	节点移动状态
1	高温	行进	1->25	16	高温	挖矿	12->12
2	高温	行进	25->24	17	沙暴	挖矿	12->12
3	晴朗	行进	24->23	18	沙暴	挖矿	12->12
4	沙暴	停留	23->23	19	高温	挖矿	12->12
5	晴朗	行进	23->22	20	高温	挖矿	12->12
6	高温	行进	22->9	21	晴朗	挖矿	12->12
7	沙暴	停留	9->9	22	晴朗	行进	12->13
8	晴朗	行进	9->15	23	高温	行进	13->15
9	高温	停留	15->15	24	晴朗	行进	15->9
10	高温	停留	15->15	25	沙暴	停留	9->9
11	沙暴	停留	15->15	26	高温	行进	9->21
12	高温	行进	15->13	27	晴朗	行进	21->27
13	晴朗	行进	13->12				
14	高温	挖矿	12->12				
15	高温	挖矿	12->12				

同理，在关卡二中，本文同样采用通过最小路径算法计算的起始点位置 54（即其起始路径为 [1,9,10,19,19,27,36,36,44,53,54]），再反复执行 Q-learning 算法求解出第二关的最优结果如表所 2 示。通过最优策略的路径求出最大收益资金为 2460 元，其剩余量见附件 Result.xlsx。

表 2 第二关该玩家穿越路径最优策略

天数	天气	行为状态	节点移动状态	天数	天气	行为状态	节点移动状态
1	高温	行进	1->9	16	高温	挖矿	55->55
2	高温	行进	9->10	17	沙暴	挖矿	55->55
3	晴朗	行进	10->19	18	沙暴	挖矿	55->55
4	沙暴	行进	19->19	19	高温	挖矿	55->55
5	晴朗	行进	19->27	20	高温	挖矿	55->55
6	高温	行进	27->36	21	晴朗	挖矿	55->55
7	沙暴	停留	36->36	22	晴朗	行进	55->62
8	晴朗	行进	36->44	23	高温	行进	62->55
9	高温	停留	44->53	24	晴朗	挖矿	55->55
10	高温	停留	53->54	25	沙暴	挖矿	55->55
11	沙暴	停留	54->54	26	高温	挖矿	55->55
12	高温	行进	54->62	27	晴朗	挖矿	55->55
13	晴朗	行进	62->55	28	晴朗	挖矿	55->55
14	高温	挖矿	55->55	29	高温	行进	55->63
15	高温	挖矿	55->55	30	高温	行进	63->64

算法的时间复杂度为 $O(n)$ ，远优于一般智能优化算法。且问题规模本身较小，本节算法运行效率相当高，宜通过重复实验搜索较优解。

5 问题二模型的建立与求解

5.1 问题描述与分析

问题二要求玩家在仅知晓当天天气的情况下决定当天的行动方案，并给出最佳行动策略。问题二在问题一的基础上使得天气情况变为随机变量，即使得天气变量分别有一定概率呈现为沙暴、高温和晴朗天气。其与第一问模型的重要不同点在于，当未来天气未知时，无法根据天气情况精确购买物资以使得物资浪费最小，且由于玩家需要优先避免被淘汰，即尽可能以最坏天气情况准备物资。针对包含随机变量的环境参数，本组改进 Q-learning 算法，设计期望最优策略，优化求解未来天气未知时的最佳行动策略。

其思维流程图如图 6 所示：

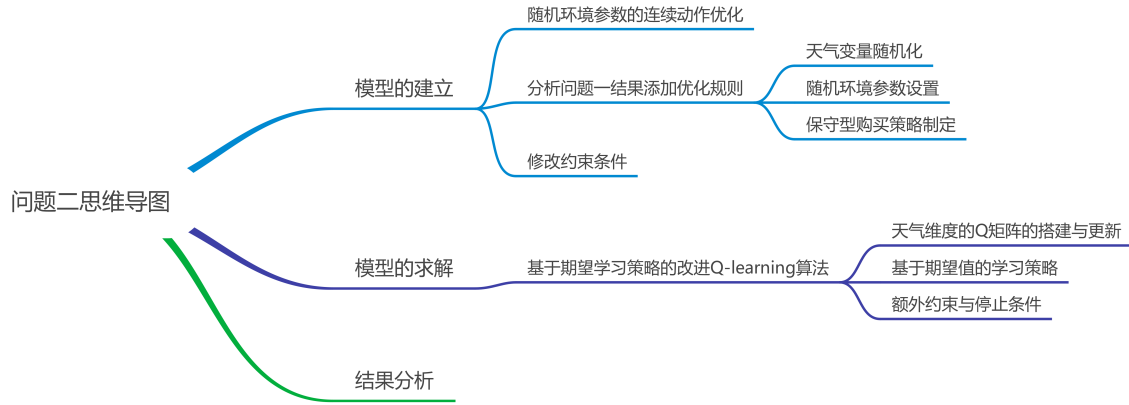


图 6 问题二思维流程图

5.2 随机环境参数的连续动作优化模型的建立

5.2.1 随机环境参数的设置

在第一问模型的基础上，将第 t 天的天气变量 $weather_t$ 变修改为随机变量如下

$$P(weather_t) = \begin{cases} p_1, & weather_t = 1, \\ p_2, & weather_t = 2, \\ p_3, & weather_t = 3. \end{cases}$$

其中 $p_{0.5}$ 、 p_1 与 $p_{1.5}$ 为事先设定的天气概率常数分别表示沙暴、高温与晴天天气的发生概率其满足

$$p_1 + p_2 + p_3 = 1.$$

此时的整体状态变量应同时包含天气变量与位置变量，即状态变量可表示为 $s_{t,weather_t}$ ，

即在状态 $s_{t,weather_t}$ 下执行动作 a_t 后即可得到 $s_{t+1,weather_{t+1}}$ 的概率为

$$P(s_{t+1,weather_{t+1}} = s_{t,weather_t} \otimes a_t) = P(weather_{t+1}|weather_t),$$

其中 $s_{t,weather_t} \otimes a_t$ 表示在状态 $s_{t,weather_t}$ 执行动作 a_t ，假设第 t 天时出现天气 $weather_t$ 的概率 $\{P(weather_t)(t = 1, 2, \dots, n)\}$ 彼此之间相互独立，即可推得

$$P(s_{t+1,weather_{t+1}} = s_{t,weather_t} \otimes a_t) = P(weather_{t+1}). \quad (13)$$

5.2.2 保守型购买策略

为优先保证玩家不会因为物资不足而导致游戏失败，购买物资时采用保守型购买策略，即准备默认之后的天气在最糟糕的情况下执行各种动作的消耗物资，具体操作可表示为

$$\begin{cases} w(0) = \sum_{t=1}^n \max_{weather_t} \Delta w(weather_t) \cdot \lambda(s_t, a_t), \\ f(0) = \sum_{t=1}^n \max_{weather_t} \Delta f(weather_t) \cdot \lambda(s_t, a_t). \end{cases}$$

即玩家将在初始点以最坏天气预算购买物资。设玩家在 $t = t_v$ 时经过村庄，此时经过采购后的物资数量可表示为

$$\begin{cases} w(t_v) = \sum_{t=t_v}^n \max_{weather_t} \Delta w(weather_t) \cdot \lambda(s_t, a_t), \\ f(t_v) = \sum_{t=t_v}^n \max_{weather_t} \Delta f(weather_t) \cdot \lambda(s_t, a_t). \end{cases} \quad (14)$$

即玩家将在村庄同样以最坏天气预算购买物资。

5.2.3 随机环境参数的连续动作优化模型

本节模型以固定环境参数的连续动作优化模型为基础，添加随机环境参数与保守购买策略约束，优化随机情况下的最佳游戏策略。整体优化模型可表示为

$$\max_{(S_{n+1}|A_n)} M_0 + \sum_{t=1}^n R(s_t, a_t), \quad (15)$$

$$\begin{aligned}
& \left\{ \begin{array}{l}
1 \leq n \leq T, \\
s_1 = s_{outset}, \\
s_{n+1} = s_{destination}, \\
|a_t| \leq weather_t, \\
\forall t = 0, 1, 2, \dots, T : w(t) \geq 0, \\
\forall t = 0, 1, 2, \dots, T : f(t) \geq 0, \\
\forall t = 0, 1, 2, \dots, T : w(t) + f(t) \leq b_0, \\
w(t+1) = w(t) - \Delta w(weather_t) \cdot \lambda(s_t, a_t), \\
f(t+1) = f(t) - \Delta f(weather_t) \cdot \lambda(s_t, a_t), \\
w(0) = \sum_{t=1}^n \max_{weather_t} \Delta w(weather_t) \cdot \lambda(s_t, a_t), \\
f(0) = \sum_{t=1}^n \max_{weather_t} \Delta f(weather_t) \cdot \lambda(s_t, a_t), \\
w(t_v) = \sum_{t=t_v}^n \max_{weather_t} \Delta w(weather_t) \cdot \lambda(s_t, a_t), \\
f(t_v) = \sum_{t=t_v}^n \max_{weather_t} \Delta f(weather_t) \cdot \lambda(s_t, a_t), \\
P(s_{t+1}, weather_{t+1} = s_t, weather_t \otimes a_t) = P(weather_{t+1}), \\
R(s_t, a_t, weather_t) = -consume(weather_t) \cdot \lambda(s_t, a_t) + income(s_t, a_t).
\end{array} \right. \quad (16)
\end{aligned}$$

5.3 基于期望学习策略的改进 Q-learning 算法

本节改进 Q-learning 算法以使其可适应带有随机变量的环境参数。首先搭建含有随机天气维度的 Q 矩阵，并将经验学习策略更变为期望形式，以求解含有随机环境参数的连续动作优化模型，具体改进如下。

5.3.1 天气维度的 Q 矩阵搭建

依照天气维度建立多重三维 Q 矩阵如图 7 所示

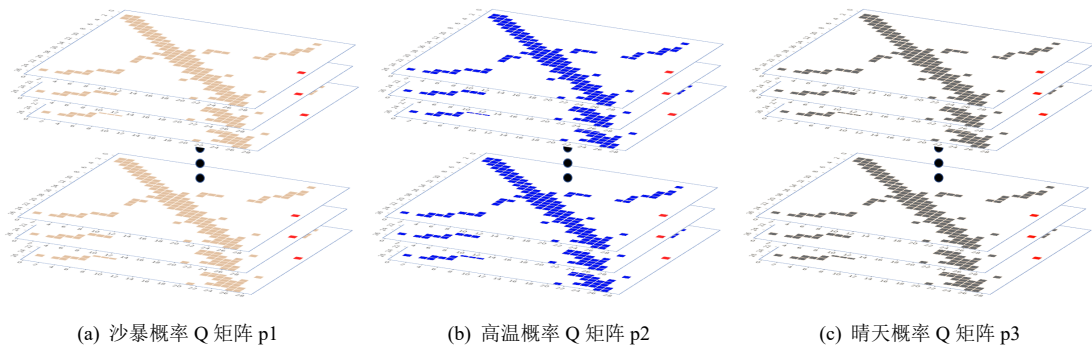


图 7 天气维度建立多重三维 Q 矩阵示意图

其中矩阵 Q_1 、 Q_2 、 Q_3 分别表示沙暴、高温、晴朗天气情况下的 Q 矩阵。即 agent 每次开始进入迭代时分别有 p_1 、 p_2 与 p_3 的概率进入矩阵 Q_1 、 Q_2 、 Q_3 的初始起点，即

$$p(s_0 \leftarrow s_{outset}(Q_j)) = p_j.$$

迭代策略沿用问题一算法中的动态 ϵ -greedy 策略，当状态变量为 $s_t(Q_i)$ 时，执行动作变量 $a_t(Q_i)$ 后得到的状态变量可表示为 $s_{t+1}(Q_j)$ ，即生成的状态变量 s_{t+1} 将以 p_j 的概率转移到矩阵 Q_j ，即

$$p(s_{t+1}(Q_j) \leftarrow s_t(Q_i) \otimes a_t(Q_i)) = p_j.$$

5.3.2 基于期望值的学习策略

假设玩家虽不知晓未来各天的精确天气，但未来天气的大致分布情况已知，即知晓未来各种天气的发生概率 p_1 、 p_2 、 p_3 。在此基础上，当玩家处于状态 $s_t(Q_i)$ 执行动作 $a_t(Q_i)$ 转移至 $s_{t+1}(Q_j)$ 时 (i 表示第 t 天时已知的天气变量， j 为表示第 $t+1$ 天时天气的随机变量)，基于期望的 Q 矩阵更新策略为

$$Q_i(s_t, a_t) \leftarrow Q_i(s_t, a_t) + \lambda[R'(s_t(Q_i), a_t(Q_i)) + \gamma \mathbb{E} \max_{a_{t+1}} Q_j(s_{t+1}, a_{t+1}) - Q_i(s_t, a_t)], \quad (17)$$

其中 λ 与 γ 分别表示学习速率与折扣因子。 $R'(s_t(Q_i), a_t(Q_i))$ 表示在矩阵 Q_i 对应的天气下，执行动作 a_t 奖励所得的奖励函数，及可通过式 4.2.4 计算，即 $R'(s_t(Q_i), a_t(Q_i)) = R(s_t, a_t, i)$ 。 $\mathbb{E} \max_{a_{t+1}} Q_j(s_{t+1}, a_{t+1})$ 表示第 $t+1$ 天可获得的最大 Q 函数值的期望，其可计算如下

$$\mathbb{E} \max_{a_{t+1}} Q_j(s_{t+1}, a_{t+1}) = \sum_{j=1}^3 p_j \cdot \max_{a_{t+1}} Q_j(s_{t+1}, a_{t+1}), \quad (18)$$

其中 $\max_{a_{t+1}} Q_j(s_{t+1}, a_{t+1})$ 表示在矩阵 Q_j 中的状态变量 s_{t+1} 下所能获得的 Q 函数最大值，进而求出最优策略路径。

5.4 实验结果及分析

第三关中，游戏过程中只有“晴朗”和“高温”天气。由算法结果可知，当玩家仅知道当天的天气状况时，最优路线一直为 [1, 4, 6, 13]。对于不同高温概率 p_2 情况下收敛所得的 Q 矩阵，重复试验 N 次后，可得通关概率 $p_{survival}$ ，并可根据不同天气分布律作出天气概率-通关率图。如图 8 所示，当沙暴概率为 0 时，agent 始终能存活，其保留资金随高温概率的增加逐渐减小。

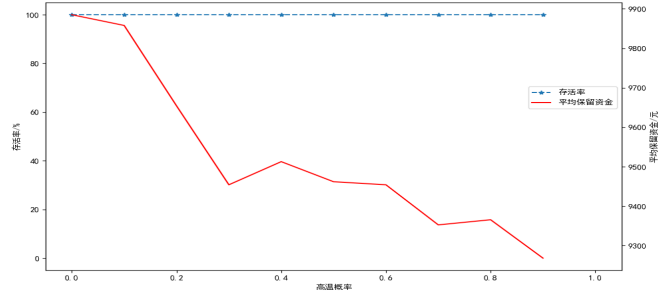
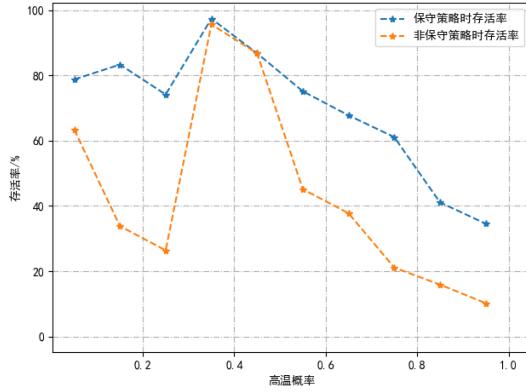
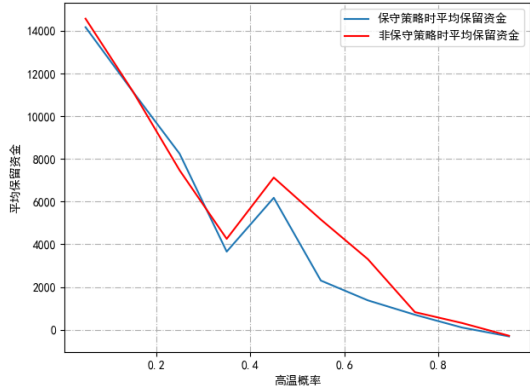


图 8 第三问学习策略结果仿真实验图

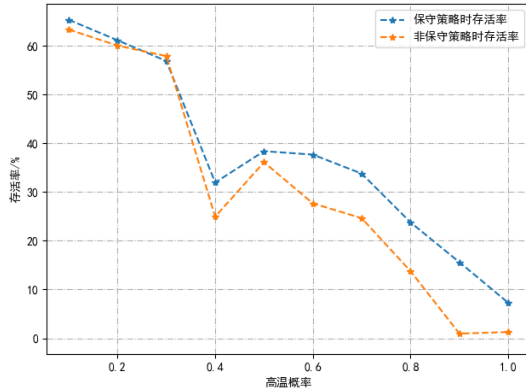
类似的，第四关中，根据附件信息可知 30 天内较少出现沙暴天气，即本组设定沙暴概率为 p_1 分别为 0.05 和 0.1，并以高温天气出现概率 p_2 作为可调环境参数。对于不同高温概率 p_2 情况下收敛所得的 Q 矩阵，以完全贪婪策略重复试验 N 次后，即可算得玩家通关概率 $p_{survival}$ 与平均保留资金 M_{mean} 。分别作出 $p_{survival} - p_2$ 图与 $M_{mean} - p_2$ 图如图 9 所示。图中的保守策略与非保守策略分别指代学习速率 λ 等于 0.8 与 1 时的学习策略。



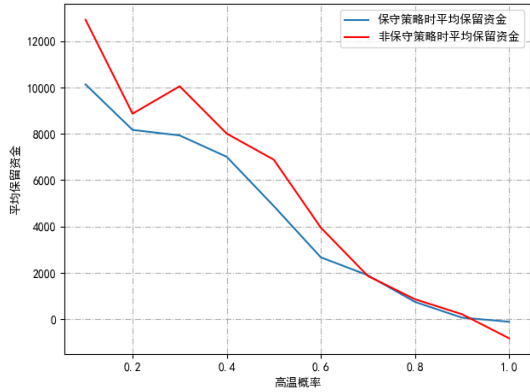
(a) 沙暴概率为 0.05 时天气与 agent 存活率关系



(b) 沙暴概率为 0.05 时天气与获得资金收益关系



(c) 沙暴概率为 0.1 时天气与 agent 存活率关系



(d) 沙暴概率为 0.1 时天气与获得资金收益关系

图 9 第四问学习策略结果仿真实验图

由图可知，通常在天气情况相同的情况下，保守策略的玩家通关概率高于非保守策略，而非保守策略的平均保留资金略高于保守策略。且随着高温天气出现概率 p_2 逐渐升高，玩家存活率与平均保留资金整体呈现出下降趋势。当纵向对比图 (a) 与图 (c)、图 (b) 与图 (d) 时，即当沙暴概率 p_1 由 0.05 增长至 0.1 时玩家通关率与保留资金都将显著下降。

6 问题三模型的建立与求解

6.1 第五关双人博弈策略模型与求解

6.1.1 第五关问题分析

鉴于在第五关中玩家需要在第 0 天制定计划且不能修改，假定游戏规则要求玩家之间构成完全竞争，即在该博弈模型中，参与人的目标是保证在整个游戏过程存活的前提下保留尽可能多的资金，资金偏少或被淘汰的一方为输家，该双玩家博弈模型是静态的完全信息变和博弈模型。因此，效用函数可以定义为资金的消耗情况。在所有路径构成的决策空间下，关于最优游戏策略的选取，我们可以得到并证明有关最优路径选择的命题。

6.1.2 完全竞争策略命题证明

命题 (最优路径命题) 选择最优路径是能保证玩家不输的最优策略。

证明 (最优路径静态博弈) 在完全信息的静态博弈中，考虑混合策略 *Nash* 平衡的博弈问题。定义效用函数 $u(n_1, n_2, \dots, n_m)$ ，满足纯策略 *Nash* 均衡

$$u(n_1^*, n_2^*, \dots, n_m^*) \geq u(n_1^*, n_2^*, \dots, n_i, \dots, n_m^*),$$

或表示为

$$\Delta u = u(n_1^*, n_2^*, \dots, n_m^*) - u(n_1^*, n_2^*, \dots, n_i, \dots, n_m^*) \geq 0,$$

其中 n_i^* 表示最优路径下第 i 个节点的最优策略。

$$n_i = \begin{cases} 1, & \text{该节点纳入路径策略,} \\ 0, & \text{该节点未纳入路径策略.} \end{cases}$$

由题意，可以根据不同情况列出策略变化导致的效用函数的变化量：

$$\Delta u = \begin{cases} (1 - \frac{1}{k})m_0, & n_i \text{ 是矿山节点且有挖矿行为} \\ \text{consume}(\text{weather}_t), & n_i \text{ 为村庄节点且有购买物资行为} \\ 15m_w + 30m_f, & n_i \text{ 为其他节点} \end{cases} \quad (19)$$

$$s.t. \quad \text{consume}(\text{weather}_t) = \begin{cases} (2k-1)(10m_w + 14m_f), & \text{weather}_t = 1, \\ (2k-1)(16m_w + 12m_f), & \text{weather}_t = 2, \end{cases} \quad (20)$$

其中， $m_w = 5$ 和 $m_f = 10$ 分别是水和食物的单价。

可知 $k = 2$ 时，该双玩家博弈模型是静态的完全信息变和博弈模型。只有当两个玩家有重叠节点时，效用函数值才会减少。故两玩家效用函数的关系可以表示为

$$u_2(n_1^{(1)}, n_2^{(1)}, \dots, n_m^{(1)}) = -\Delta u_1(n_1^{(1)}, n_2^{(1)}, \dots, n_m^{(1)}).$$

根据效用函数关系，有

$$u_1(n_1^*, n_2^*, \dots, n_m^*) \geq u_2(n_1^*, n_2^*, \dots, n_m^*) \geq u_2(n_1^{(1)}, n_2^{(1)}, \dots, n_m^{(1)}), \quad (21)$$

其中， $n_i^{(1)}$ 代表路径变化后玩家采取的策略。

可知，当 $k = 2$ 时，双玩家博弈模型中，效用函数在参与人的策略空间中的最值在 *Nash* 均衡的极值处，即最优路径序列处取得，保证了资金花费最少。因此，选择最优路径是能保证玩家不输的最优策略。

基于问题一优化算法可解得最优路径有且仅有两条，分别为 $[1, 4, 6, 13]$ 与 $[1, 5, 6, 13]$ 。当两个玩家选择相同最优路径时，各自收益均为 7640 元，当两个玩家选取不同最优路径时，各自收益均为 8840 元。结果示意图表示如下

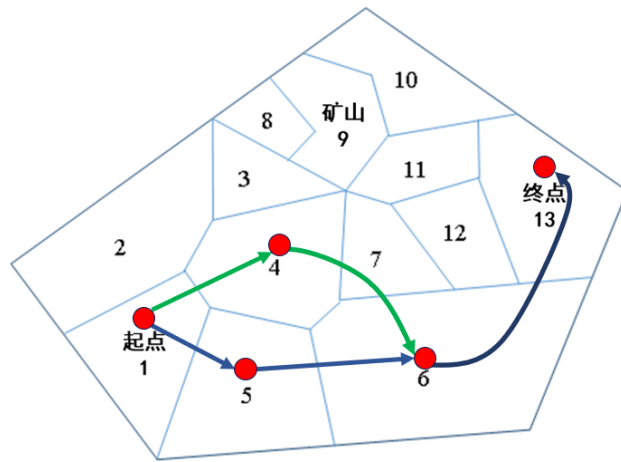


图 10 第五关双玩家策略路径图

6.2 第六关三人合作策略与求解

6.2.1 第六关问题分析

鉴于在第六关中玩家不知道未来的天气状况，但知晓其余玩家当天的行动方案和剩余的资源数量。若在此情况下执行竞争策略，问题将转化为三人博弈模型，此时纳什均衡点计算难度将大幅度上升，且由于天气情况未知，执行竞争策略时玩家由于生存物资不足而被淘汰的概率将大幅度上升。假定玩家为保证自身的通关概率自发进行合作，对其他玩家建模并自发采取谦让策略。

6.2.2 谦让策略

每个玩家在问题二中第四关模型与求解所得的最优策略模型对自身与其他玩家进行建模，计算第 n 号玩家 ($n = 1, 2, 3$) 当前的资源总量

$$V_n(t) = Valve(w_n(t)) + Valve(f_n(t)) + M_n(t), \quad (22)$$

其中 $w_n(t)$ 、 $f_n(t)$ 与 $M_n(t)$ 分别表示第 n 号玩家第 t 天的剩余饮水、剩余食物与剩余资金， $Valve(w_n(t))$ 与 $Valve(f_n(t))$ 分别表示剩余饮水、剩余食物的折价。

在第 t 天时，此时三人默认资源总量最少的玩家将优先执行当前最优策略，即其选择 Q 矩阵中 Q 函数最大的策略，即 $n_{best} = \arg_n \max V_n(t)$ 、 $n_{worst} = \arg_n \min V_n(t)$ 剩余的为玩家为 n_{middle} 。之后多个玩家处于同一位置变量 s_t 时，将由执行策略的优先度为 $n_{best} > n_{middle} > n_{worst}$ ，即将由优先度高的玩家优先执行 Q 值更高的动作变量 a_t ，且优先度较低的玩家不能选择优先度高的玩家以选择的相同的动作变量 a_t 。设定沙暴概率 $p_1 = 0.05$ ，以谦让策略重复实验 N 次后，算得玩家通关概率 $p_{n,survival}$ 与平均保留资金 $M_{n,mean}$ 如图 11 所示

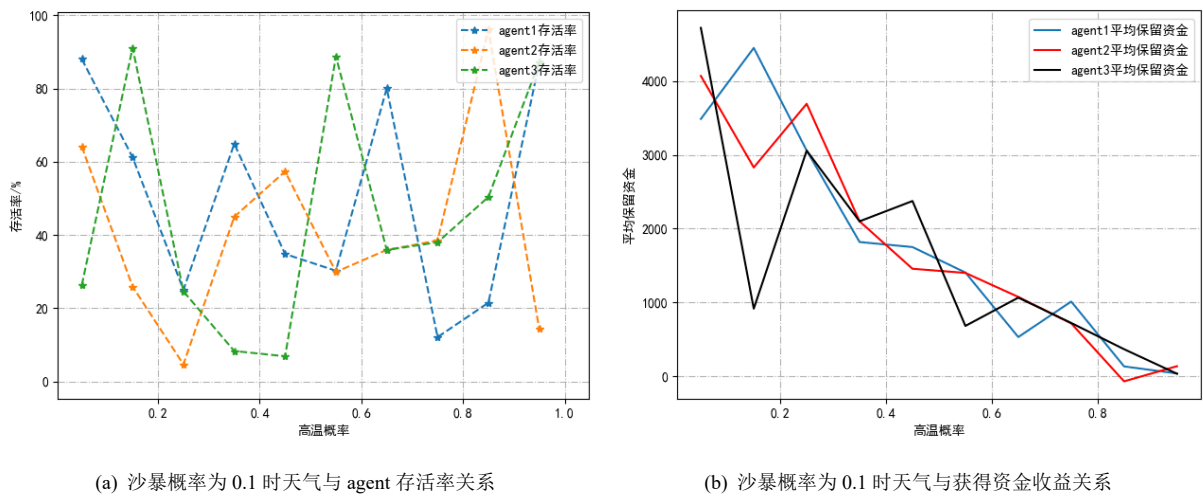


图 11 谦让策略结果仿真实验图

图 11 (a) 中表示执行谦让策略后各玩家的通关概率，其远高于三者同时执行个人最优优化策略时的通关概率。三人的平均保留资金将随高温天气出现概率的升高而降低，复合本组的预期。

7 模型的评价

7.1 模型的优点

- (1) 本文中 Q-learning 算法的时间复杂度为 $O(n)$ ，远优于一般群集智能优化算法（其时间复杂度为 $O(n^2) - O(n^3)$ ）。其算法运行速度较快，并且每个 agent 可通过不同的策略自行探索动态学习，无需训练集也对初始解没有要求，鲁棒性较强。
- (2) 基于期望学习策略的 Q-learning 算法可适应带有随机变量的环境参数，通过天气概率计算奖励的期望值进行 Q 矩阵的更新策略。

7.2 模型的缺点

Q-learning 存在过高估计的问题，学习策略易陷入局部最优解，需要反复运行算法以借助 ϵ -greedy 策略的探索特性跳出局部最优解。

7.3 改进与展望

在多人博弈策略中，Q-learning 算法的每个 agent 能设置一种动态中心管理策略，从而实现对未知环境下的全局动态调度情况，并可通过改变策略使其跳出局部最优解。

参考文献

- [1] Wang S, Lin W. Planning of Opposite Q Learning Based on Virtual Sub-Target in Unknown Environment[C]//2018 International Conference on Network, Communication, Computer Engineering (NCCE 2018). Atlantis Press, 2018.
- [2] Feng C, Sun M, Zhang J. Reinforced Deterministic and Probabilistic Load Forecasting via Q-Learning Dynamic Model Selection[J]. IEEE Transactions on Smart Grid, 2019, 11(2): 1377-1386.
- [3] 李邦云. 基于 Bellman-Ford 算法的配电网节能控制研究 [J]. 舰船电子工程, 2018, 38(8): 37-41.
- [4] 郑延斌, 樊文鑫, 韩梦云, 等. 基于博弈论及 Q 学习的多 Agent 协作追捕算法 [J]. 计算机应用, 2020: 0-0.

附录 A 沙漠游戏 Q-learning 算法源代码

A.1 模型预处理

模型预处理，每一关的图矩阵构造和天气的记录，用于构造 Q 矩阵

```
1 import numpy as np
2 from numpy import isnan
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5
6 def get_maze(checkpointnum):
7     """
8     :param checkpointnum: 关卡数目
9     :return: 迷宫, 天气
10    """
11    maze = []
12    weather = [] # 1晴天, 2高温, 3沙暴
13    assert checkpointnum in list(range(1,7))
14    if checkpointnum==1:
15        weather = [2,2,1,3,1,2,3,1,2,2,
16                  3,2,1,2,2,2,3,3,2,2,
17                  1,1,2,1,3,2,1,1,2,2]
18        maze = np.full((27+1, 28+1), np.nan) # 28为挖, +1为0的地方不要
19        maze[12,28] = 0 # 挖矿
20        for i in range(1, 27+1): # 停一天
21            maze[i, i] = 0
22        # 临边
23        maze[1, 25],maze[1, 2]=0,0
24        maze[2, 1],maze[2, 3]=0,0
25        maze[3, 2],maze[3, 25],maze[3, 4]=0,0,0
26        maze[4, 3],maze[4, 25],maze[4, 24],maze[4, 5]=0,0,0,0
27        maze[5, 4],maze[5, 24],maze[5, 6]=0,0,0
28        maze[6, 5],maze[6, 24],maze[6, 23],maze[6, 7]=0,0,0,0
29        maze[7, 6],maze[7, 22],maze[7, 8]=0,0,0
30        maze[8, 7],maze[8, 22],maze[8, 9]=0,0,0
31        maze[9, 8],maze[9, 22],maze[9, 21],maze[9, 17],maze[9,
```

```

16],maze[9, 15],maze[9, 10]=0,0,0,0,0,0,0
32 maze[10, 9],maze[10, 15],maze[10, 13],maze[10, 11]=0,0,0,0
33 maze[11, 10],maze[11, 13],maze[11, 12]=0,0,0
34 maze[12, 11],maze[12, 13],maze[12, 14]=0,0,0
35 maze[13, 10],maze[13, 15],maze[13, 14],maze[13, 12],maze[13,
    11]=0,0,0,0,0
36 maze[14, 15],maze[14, 16],maze[14, 12],maze[14, 13]=0,0,0,0
37 maze[15, 10],maze[15, 9],maze[15, 16],maze[15, 14],maze[15,
    13]=0,0,0,0,0
38 maze[16, 14],maze[16, 15],maze[16, 9],maze[16, 17],maze[16,
    18]=0,0,0,0,0
39 maze[17, 9],maze[17, 21],maze[17, 18],maze[17, 16]=0,0,0,0
40 maze[18, 17],maze[18, 20],maze[18, 19],maze[18, 16]=0,0,0,0
41 maze[19, 18],maze[19, 20]=0,0
42 maze[20, 21],maze[20, 18],maze[20, 19]=0,0,0
43 maze[21, 27],maze[21, 23],maze[21, 22],maze[21, 9],maze[21,
    17],maze[21, 20]=0,0,0,0,0,0
44 maze[22, 7],maze[22, 23],maze[22, 21],maze[22, 9],maze[22,
    8]=0,0,0,0,0
45 maze[23, 24],maze[23, 26],maze[23, 21],maze[23, 22],maze[23,
    6]=0,0,0,0,0
46 maze[24, 5],maze[24, 6],maze[24, 23],maze[24, 26],maze[24,
    25],maze[24, 4]=0,0,0,0,0,0
47 maze[25, 1],maze[25, 24],maze[25, 3],maze[25, 4],maze[25,
    26]=0,0,0,0,0
48 maze[26, 25],maze[26, 24],maze[26, 23],maze[26, 27]=0,0,0,0
49 maze[27, 26],maze[27, 21]=0,0
50 pass
51 elif checkpointnum==2:
52     weather = [2, 2, 1, 3, 1, 2, 3, 1, 2, 2,
53                 3, 2, 1, 2, 2, 2, 3, 3, 2, 2,
54                 1, 1, 2, 1, 3, 2, 1, 1, 2, 2]
55     maze = np.full((64 + 1, 65 + 1), np.nan)
56     maze[30,65] = 0    # 挖矿
57     maze[55,65] = 0    # 挖矿

```

```

58     for i in range(1, 64+1):
59         maze[i, i] = 0 # 停一天
60         paishu = int((i-1)/8)+1 # 第几排
61         tou = (paishu - 1) * 8 + 1 # 头
62         wei = paishu * 8 # 尾
63         if paishu==1:
64             if i==tou:
65                 maze[tou,tou+1],maze[tou,tou+8]=0,0
66             elif i==wei:
67                 maze[wei,wei-1],maze[wei,wei+7],maze[wei,wei+8]=0,0,0
68             else:
69                 maze[i,i-1],maze[i,i+1],maze[i,i+7],maze[i,i+8]=0,0,0,0
70         if paishu==8:
71             if i==tou:
72                 maze[tou,tou-8],maze[tou,tou-7],maze[tou,tou+1]=0,0,0
73             elif i==wei:
74                 maze[wei,wei-1],maze[wei,wei-8]=0,0
75             else:
76                 maze[i,i-1],maze[i,i+1],maze[i,i-8],maze[i,i-7]=0,0,0,0
77         if paishu in [2,4,6]:
78             if i==tou:
79                 maze[tou,tou-8],maze[tou,tou-7],maze[tou,tou+1],maze[tou,tou+8],maze
80             elif i==wei:
81                 maze[wei,wei-1],maze[wei,wei-8],maze[wei,wei+8]=0,0,0
82             else:
83                 maze[i,i-1],maze[i,i+1],maze[i,i-8],maze[i,i-7],maze[i,i+8],maze[i,
84         if paishu in [3,5,7]:
85             if i==tou:
86                 maze[tou,tou+1],maze[tou,tou-8],maze[tou,tou+8]=0,0,0
87             elif i==wei:
88                 maze[wei,wei-1],maze[wei,wei-9],maze[wei,wei-8],maze[wei,wei+8],maze
89             else:
90                 maze[i,i-1],maze[i,i+1],maze[i,i-8],maze[i,i-9],maze[i,i+8],maze[i,
91         pass
92     elif checkpointnum==3:

```

```

93     weather = []
94     maze = np.full((13 + 1, 14 + 1), np.nan)
95     maze[9, 14] = 0 # 挖矿
96     for i in range(1, 13 + 1):
97         maze[i, i] = 0 # 停一天
98     maze[1, 2], maze[1, 5], maze[1, 4] = 0, 0, 0
99     maze[2, 1], maze[2, 4], maze[2, 3] = 0, 0, 0
100    maze[3, 2], maze[3, 4], maze[3, 9], maze[3, 8] = 0, 0, 0, 0
101    maze[4, 1], maze[4, 2], maze[4, 3], maze[4, 7], maze[4, 6], maze[4,
        5] = 0, 0, 0, 0, 0, 0
102    maze[5, 1], maze[5, 4], maze[5, 6] = 0, 0, 0
103    maze[6, 5], maze[6, 4], maze[6, 7], maze[6, 12], maze[6, 13] = 0, 0, 0, 0, 0
104    maze[7, 4], maze[7, 11], maze[7, 12], maze[7, 6] = 0, 0, 0, 0
105    maze[8, 3], maze[8, 9] = 0, 0
106    maze[9, 3], maze[9, 8], maze[9, 11], maze[9, 10] = 0, 0, 0, 0
107    maze[10, 9], maze[10, 11], maze[10, 13] = 0, 0, 0
108    maze[11, 9], maze[11, 10], maze[11, 13], maze[11, 12], maze[11,
        7] = 0, 0, 0, 0, 0
109    maze[12, 7], maze[12, 11], maze[12, 13], maze[12, 6] = 0, 0, 0, 0
110    maze[13, 6], maze[13, 12], maze[13, 11], maze[13, 10] = 0, 0, 0, 0
111    pass
112 elif checkpointnum==4:
113     weather = []
114     maze = np.full((25 + 1, 26 + 1), np.nan)
115     maze[18, 26] = 0 # 挖矿
116     for i in range(1, 25 + 1):
117         maze[i, i] = 0 # 停一天
118         if i==1:
119             maze[i, i+1], maze[i, i+5] = 0, 0
120         elif i==5:
121             maze[i, i-1], maze[i, i+5] = 0, 0
122         elif i==21:
123             maze[i, i+1], maze[i, i-5] = 0, 0
124         elif i==25:
125             maze[i, i-1], maze[i, i-5] = 0, 0

```



```

126         elif i in [2,3,4]:
127             maze[i,i-1],maze[i,i+1],maze[i,i+5]=0,0,0
128         elif i in [22,23,24]:
129             maze[i,i-1],maze[i,i+1],maze[i,i-5]=0,0,0
130         elif i in [6,11,16]:
131             maze[i,i-5],maze[i,i+1],maze[i,i+5]=0,0,0
132         elif i in [10,15,20]:
133             maze[i,i-5],maze[i,i-1],maze[i,i+5]=0,0,0
134         else:
135             maze[i, i-1], maze[i, i + 1], maze[i, i-5], maze[i, i+5] =
                0, 0, 0, 0
136     pass
137 elif checkpointnum==5:
138     weather = [1,2,1,1,1,1,2,2,2,2,]
139     maze = np.full((13 + 1, 14 + 1), np.nan)
140     maze[9, 14] = 0 # 挖矿
141     for i in range(1, 13 + 1):
142         maze[i, i] = 0 # 停一天
143     maze[1, 2],maze[1, 5],maze[1, 4]=0,0,0
144     maze[2, 1],maze[2, 4],maze[2, 3]=0,0,0
145     maze[3, 2],maze[3, 4],maze[3, 9],maze[3, 8]=0,0,0,0
146     maze[4, 1],maze[4, 2],maze[4, 3],maze[4, 7],maze[4, 6],maze[4,
        5]=0,0,0,0,0,0
147     maze[5, 1],maze[5, 4],maze[5, 6]=0,0,0
148     maze[6, 5],maze[6, 4],maze[6, 7],maze[6, 12],maze[6, 13]=0,0,0,0,0
149     maze[7, 4],maze[7, 11],maze[7, 12],maze[7, 6]=0,0,0,0
150     maze[8, 3],maze[8, 9]=0,0
151     maze[9, 3],maze[9, 8],maze[9, 11],maze[9, 10]=0,0,0,0
152     maze[10, 9],maze[10, 11],maze[10, 13]=0,0,0
153     maze[11, 9],maze[11, 10],maze[11, 13],maze[11, 12],maze[11,
        7]=0,0,0,0,0
154     maze[12, 7],maze[12, 11],maze[12, 13],maze[12, 6]=0,0,0,0
155     maze[13, 6],maze[13, 12],maze[13, 11],maze[13, 10]=0,0,0,0
156     pass
157 elif checkpointnum==6:

```

```

158     weather = []
159     maze = np.full((25 + 1, 26 + 1), np.nan)
160     maze[18, 26] = 0 # 挖矿
161     for i in range(1, 25 + 1):
162         maze[i, i] = 0 # 停一天
163         if i==1:
164             maze[i,i+1],maze[i,i+5]=0,0
165         elif i==5:
166             maze[i,i-1],maze[i,i+5]=0,0
167         elif i==21:
168             maze[i,i+1],maze[i,i-5]=0,0
169         elif i==25:
170             maze[i,i-1],maze[i,i-5]=0,0
171         elif i in [2,3,4]:
172             maze[i,i-1],maze[i,i+1],maze[i,i+5]=0,0,0
173         elif i in [22,23,24]:
174             maze[i,i-1],maze[i,i+1],maze[i,i-5]=0,0,0
175         elif i in [6,11,16]:
176             maze[i,i-5],maze[i,i+1],maze[i,i+5]=0,0,0
177         elif i in [10,15,20]:
178             maze[i,i-5],maze[i,i-1],maze[i,i+5]=0,0,0
179         else:
180             maze[i, i-1], maze[i, i + 1], maze[i, i-5], maze[i, i+5] =
181                 0, 0, 0, 0
182     pass
183 else:
184     print("输入关卡数错误")
185     return None
186 return maze, np.array(weather)
187
188 def is_duicheng(maze, num):
189     maze = maze[:num, :num]
190     print(maze.shape)
191     for i in range(num):

```

```

192     for j in range(num):
193         if maze[i,j]!=maze[j,i] and not isnan(maze[j,i]):
194             print("不是对称",i,j)
195
196
197
198
199 if __name__ == "__main__":
200     maze,weather = get_maze(1)
201     print(maze.shape)
202     print(maze)
203     is_duicheng(maze,maze.shape[0])
204     # https://blog.csdn.net/qq\_42554007/article/details/82624418
205     cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True,dark=0.6,
        light=0.6)
206     sns.heatmap(maze, linewidths=0.05, cbar = True, cmap=cmap) #
207     plt.show()

```

A.2 第一关代码

第一关 Q-learning 学习代码，输出序列为 agent 路径 [1, 25, 24, 23, 23, 22, 9, 9, 15, 15, 15, 15, 13, 12, 28, 28, 28, 28, 28, 28, 28, 28, 13, 15, 9, 9, 21, 27]

```

1 import numpy as np
2 from maze import get_maze
3 import random
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 guanqia = 1 #关卡
7 learn_num = 100000 #迭代次数
8 yuzhi = int(learn_num*0.9) # 策略域
9 W_UP = 1200 #负重上线
10 aleph, gamma = 0.8, 0.8 #学习率
11 shouyi = 1000 # 挖矿收益
12 cunzhuang = [15] # 村庄
13 kuang = [12] # 矿山

```

```

14 ISREAD_MAZES = False
15 maze, weathers = get_maze(guanqia)
16 wakuang = maze.shape[0] # 终点位置
17 PATH = 'demo1'
18 #[25, 24, 23, 23, 22, 9, 9, 15, 13, 12, 28, 28, 28, 28, 28, 14, 14, 14,
19     14, 14, 15, 9, 21, 27]
20
21 # starts_path= [1]
22 # starts_A= []
23 starts_path = [1,25, 24, 23, 23, 22, 9, 9, 15, 15, 15, 15, 13, 12 , 28,
24     28, 28, 28, 28, 28, 28, 28, 13,15,9, 9,21] # 开始路径
25 starts_A = [25, 24, 23, 23, 22, 9, 9, 15, 15, 15, 15, 13, 12 , 28, 28,
26     28, 28, 28, 28, 28, 28, 13,15,9, 9,21] # 开始状态
27
28 # starts_path = [1,25, 24, 23, 23, 22, 9, 9, 15, 13,
29     12,12,12,12,12,12,12,12,12,13,15, 9, 21,] # 开始路径
30 # starts_A = [25, 24, 23, 23, 22, 9, 9, 15, 13,
31     12,12,28,28,28,28,28,28,28,13,15, 9, 21,] # 开始状态
32 SSSSS = starts_path[len(starts_path) - 1]# 起点
33
34
35
36
37
38
39
40
41
42 def get_pathmap(maze):
43     path_map = []
44     # 每个状态拥有的动作
45     for i in range(maze.shape[0]): # 初始化所有可行域
46         temp = []
47         for j in range(1, maze.shape[1]):
48             if not np.isnan(maze[i,j]):
49                 temp.append(j)
50         path_map.append(temp)
51     return path_map
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

44     w_i = 3*5+2*7
45     p_i = 5*5+10*7
46 elif weather==2:
47     w_i = 3*8+2*6
48     p_i = 5*8+10*6
49 else:
50     w_i = 3*10+2*10
51     p_i = 5*10+10*10
52 return w_i, p_i
53
54
55 def probability_fun(sss, max_a, probability=0.8, k=None):
56     '''
57     :param probability: 0为贪婪, 1为随机策略, 其他为sigema策略
58     '''
59     if probability==1:
60         return random.choice(sss)
61     elif probability==0:
62         list_a = max_a.tolist()
63         q_max = float('-inf')
64         for i in sss:
65             if list_a[i] > q_max and not np.isnan(list_a[i]):
66                 q_max=list_a[i]
67         # if probability==0:
68         #     print(q_max, list_a.index(q_max))
69         return list_a.index(q_max)
70     else:
71         if k<yuzhi:
72             probability=-(1-0.5)*k/yuzhi+1
73         else:
74             probability=0
75         if np.random.random()<probability:
76             return random.choice(sss)
77         else:
78             list_a = max_a.tolist()

```

```

79         q_max = float('-inf')
80         for i in sss:
81             if list_a[i] > q_max and not np.isnan(list_a[i]):
82                 q_max = list_a[i]
83         # if probability==0:
84         #     print(q_max, list_a.index(q_max))
85         return list_a.index(q_max)
86
87
88 def get_W_status(starts_path, starts_A):
89     s = 1 # 起点
90     w = 0 # 背包负重
91     w_chunzhuang = 0 # 村庄买的东西
92     plan = [W_UP, 0] # 计划购买栈
93     Rs = []
94     flag = False # 是否饿死
95     for t in range(len(starts_path)-1):# 第0天就开始动, 第30天没有动
96         where = starts_path[t]
97         s_p = starts_A[t] if starts_A[t] != wakuang else
98             where#下一步要干嘛
99
100         w_i, r_i = get_w_i(weathers[t]) # 第t天的基础消耗
101         R = -r_i # 基础消耗
102         w_i_i = w_i # 基础消耗
103         if where!=s_p: # 走, 2倍
104             R = - 2 * r_i
105             w_i_i = 2 * w_i
106         if s_p == wakuang: # 挖矿钱
107             R = -3 * r_i + shouyi
108             w_i_i = 3 * w_i
109         w = w + w_i_i
110         w_chunzhuang = w_chunzhuang + w_i_i
111
112         if plan[0] >= w_i_i: # 不需要村庄
113             plan[0] = plan[0] - w_i_i

```

```

113     else: # 需要用村庄
114         plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
115         plan[0] = 0
116         if where!=s_p: # 走, 2倍
117             R = - 4 * r_i
118         if s_p == wakuang: # 挖矿钱
119             R = -6 * r_i + shouyi # 挖矿钱
120         if plan[1] < 0:
121             flag = True # 死了
122
123     print("第{}天执行{}->{}".format(t,where,s_p), plan,
124           w_chunzhuang,w)
125
126     # 村庄买东西
127     if where in cunzhuang: # 村庄
128         plan[1] = plan[1] + w_chunzhuang
129         w_chunzhuang = 0
130
131     Rs.append(R)
132     return plan, Rs, flag
133
134 def get_path(mazes, path_map):
135     path = starts_path.copy()
136     A = starts_A.copy()
137     s = SSSSS
138     for t in range(start_t, len(weathers)-1): # (10, 30)
139         map = mazes[t,:,:]
140         if s==wakuang - 1:
141             break
142         a = probability_fun(path_map[s], map[s,:], 0)
143         s_p = a if a!=wakuang else s
144
145         path.append(s_p)
146         A.append(a)

```

```

147     s = s_p
148     return path, A
149
150
151 if __name__ == '__main__':
152     mazes = []      # 时间图Q
153     for i in range(len(weathers)):
154         mazes.append(maze)
155     mazes = np.array(mazes).copy()
156     path_map = get_pathmap(maze)
157     print(mazes.shape) # mazes (时间, 状态, 动作)
158     print(path_map)
159     print(path_map[21])
160
161     # R放入初值中
162     # for t in range(len(weathers)):
163     #     for j in range(wakuang): # 初始化所有可行域
164     #         for a in path_map[j]:
165     #             w_i, r_i = get_w_i(weathers[t])
166     #             R = -r_i
167     #             if a != j: # 走, 2倍
168     #                 R = - 2 * r_i
169     #             if a == maze.shape[1] - 1:
170     #                 R = -3 * r_i + shouyi # 挖矿钱
171     #             mazes[t,j,a] = R
172
173     # mazes付初值
174     if ISREAD_MAZES:
175         mazes =
176
177         np.array(np.load("./data/{_}_{_}.npz".format(PATH,guanqia))['data'])
178
179     # 初值的可视化
180     cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True, dark=0.7,
181                                  light=0.3)
182     sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True, cmap=cmap) #

```



```

180 plt.show()
181
182 print('##### Q-learning #####')
183 planssss, Rs, flag = get_W_status(starts_path, starts_A)
184 print(flag, planssss)
185 start_t = len(starts_path)-1
186 print("起点", SSSSS, "第几天", start_t, len(weathers)-1, "矿位置",
      wakuang)
187 for k in range(learn_num):# 迭代次数
188     s = SSSSS # 起点
189
190     plan = planssss.copy() # 计划购买栈
191     path = starts_path.copy()
192     w = 1200-planssss[0] # 背包负重
193     w_chunzhuang = 1200-planssss[0] # 村庄买的东西
194     A = starts_A.copy()
195     flag = False
196     # print("*"*50)
197     # is_kuang = False
198     # print(start_t, len(weathers))
199     for t in range(start_t, len(weathers)-1):#(10, 30)
200         a = probability_fun(path_map[s], mazes[t, s, :], k=k) # 动作
201         # if s not in kuang and a in kuang:
202         #     is_kuang=True
203         # if is_kuang:
204         #     a=s
205         #     is_kuang = False
206
207         s_p = a if a != wakuang else s # 下一步要干嘛
208
209         w_i, r_i = get_w_i(weathers[t])
210         R = -r_i # 走一步消耗
211         w_i_i = w_i
212         if a != s: # 走, 2倍
213             R = - 2 * r_i

```

```

214         w_i_i = 2 * w_i
215     if a == wakuang:
216         R = -3 * r_i + shouyi # 挖矿钱
217         w_i_i = 3 * w_i
218     if weathers[t] == 3: # 沙包必须停留
219         s_p = s
220         # if a!=wakuang and a!=s:
221         #     a=s
222
223     w = w + w_i_i
224     w_chunzhuang = w_chunzhuang + w_i_i
225
226     if plan[0] >= w_i_i: # 不需要村庄
227         plan[0] = plan[0] - w_i_i
228     else: # 需要用村庄
229         plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
230         plan[0] = 0
231         if s != s_p: # 走, 2倍
232             R = - 4 * r_i
233         if a == wakuang: # 挖矿钱
234             R = -6 * r_i + shouyi # 挖矿钱
235         if plan[1] < 0:
236             flag = True # 死了
237     if plan[1] < 0:
238         flag = True # 死了
239
240     # print("第{}天执行{}, {}->{}".format(t,a,s,s_p), plan)
241
242     # 村庄买东西
243     if s in cunzhuang: # 村庄
244         plan[1] = plan[1] + w_chunzhuang
245         w_chunzhuang = 0
246
247     max_q = mazes[t + 1, s_p, probability_fun(path_map[s_p],
        mazes[t + 1, s_p, :], 0)] # 找到最大位置的q值

```

```

248
249     if s_p == wakuang - 1: # 终点
250         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
251         s = s_p # 更新位置
252         # print(path)
253         break # 29
254     elif flag: # 中止条件
255         R = R - 1000000
256         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
257             + gamma * max_q)
258         s = s_p # 更新位置
259         break
260     elif t == len(weathers) - 1 - 1 and s_p != wakuang - 1:
261         R = R - 1000000
262         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
263             + gamma * max_q)
264         s = s_p # 更新位置
265         break
266     else:
267         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
268             + gamma * max_q)
269         s = s_p # 更新位置
270
271     A.append(a)
272     path.append(s)
273     # break
274
275     np.savez_compressed("./data/{}_{}.npz".format(PATH, guanqia),
276         data=mazes)
277
278     path, A = get_path(mazes, path_map)
279     print(path)
280     print(A)
281     #
282     #

```

```

279     # for t in range(start_t, len(weathers)):
280     #     map = mazes[t,:,:]
281     #     for i in range(map.shape[0]):
282     #         for j in range(map.shape[1]):
283     #             if map[i,j]<-100000 and not np.isnan(map[i,j]):
284     #                 map[i, j] = -1000000
285     #     cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True,
286     #                                   dark=0.7, light=0.3)
287     #     sns.heatmap(map, linewidths=0.05, cbar=True, cmap=cmap) #
288     #     plt.show()

```

A.3 第二关代码

第二关 Q-learning 学习代码，输出序列为 agent 路径 [1, 9, 10, 19, 19, 27, 36, 36, 44, 53, 54, 54, 55, 55, 55, 46, 39, 39, 39, 30, 30, 30, 30, 30, 30, 30, 39, 47, 56, 64]

```

1  import numpy as np
2  from maze import get_maze
3  import random
4  import matplotlib.pyplot as plt
5  import seaborn as sns
6  guanqia = 2 #关卡
7  learn_num = 1000000 #迭代次数
8  yuzhi = int(learn_num*0.9) # 策略域
9  W_UP = 1200 #负重上线
10 aleph, gamma = 1, 0.8 #学习率
11 shouyi = 1000 # 挖矿收益
12 cunzhuang = [39,62] # 村庄
13 kuang = [30,55] # 矿山
14 ISREAD_MAZES = False
15 maze, weathers = get_maze(guanqia)
16 wakuang = maze.shape[0] # 终点位置
17 PATH = 'demo2'
18 # [1,9, 10, 19, 19, 27, 36, 36, 44, 53, 54, 54, 55, 65, 65, 65, 62, 62,
19     62, 55, 65, 65, 65, 65, 65, 65, 65, 65, 56, 64]

```

```

20 # starts_path=
    [1,9,10,19,19,27,36,36,44,53,54,62,55,55,55,55,55,55,55,62,55,55]
21 # starts_A=
    [9,10,19,19,27,36,36,44,53,54,62,55,65,65,65,65,65,65,65,62,55,65]
22 # starts_path= [1, 2, 10, 11, 11, 12, 13, 13, 22, 30]
23 # starts_A= [ 2, 10, 11, 11, 12, 13, 13, 22, 30]
24 #
25 starts_path= [1, 9, 10, 19, 19, 27, 36, 36, 44, 53, 54, 54, 62,
26               55, 65, 65, 65, 65, 65, 65, 65,65, 62, 55, 65, 65, 65, 65,
                65, 56, 64]
27 starts_A = [9, 10, 19, 19, 27, 36, 36, 44, 53, 54, 54, 62, 55,
28             65, 65, 65, 65, 65, 65, 65,65, 62, 55, 65, 65, 65, 65, 65,
                56, 64]
29
30 SSSSS = starts_path[len(starts_path) - 1]# 起点
31
32
33 def get_pathmap(maze):
34     path_map = []
35     # 每个状态拥有的动作
36     for i in range(maze.shape[0]): # 初始化所有可行域
37         temp = []
38         for j in range(1, maze.shape[1]):
39             if not np.isnan(maze[i,j]):
40                 temp.append(j)
41         path_map.append(temp)
42     return path_map
43
44
45 def get_w_i(weather):
46     if weather==1:
47         w_i = 3*5+2*7
48         p_i = 5*5+10*7
49     elif weather==2:
50         w_i = 3*8+2*6

```

```

51     p_i = 5*8+10*6
52 else:
53     w_i = 3*10+2*10
54     p_i = 5*10+10*10
55     return w_i, p_i
56
57
58 def probability_fun(sss, max_a, probability=0.8, k=None):
59     '''
60     :param probability: 0为贪婪, 1为随机策略, 其他为sigema策略
61     '''
62     if probability==1:
63         return random.choice(sss)
64     elif probability==0:
65         list_a = max_a.tolist()
66         q_max = float('-inf')
67         for i in sss:
68             if list_a[i] > q_max and not np.isnan(list_a[i]):
69                 q_max=list_a[i]
70         # if probability==0:
71         #     print(q_max, list_a.index(q_max))
72         return list_a.index(q_max)
73     else:
74         if k<yuzhi:
75             probability=-(1-0.5)*k/yuzhi+1
76         else:
77             probability=0
78         if np.random.random()<probability:
79             return random.choice(sss)
80         else:
81             list_a = max_a.tolist()
82             q_max = float('-inf')
83             for i in sss:
84                 if list_a[i] > q_max and not np.isnan(list_a[i]):
85                     q_max = list_a[i]

```

```

86         # if probability==0:
87         #     print(q_max, list_a.index(q_max))
88         return list_a.index(q_max)
89
90
91 def get_W_status(starts_path, starts_A):
92     s = 1 # 起点
93     w = 0 # 背包负重
94     w_chunzhuang = 0 # 村庄买的东西
95     plan = [W_UP, 0] # 计划购买栈
96     Rs = []
97     flag = False # 是否饿死
98     for t in range(len(starts_path)-1):# 第0天就开始动, 第30天没有动
99         where = starts_path[t]
100         s_p = starts_A[t] if starts_A[t] != wakuang else
            where#下一步要干嘛
101
102         w_i, r_i = get_w_i(weathers[t]) # 第t天的基础消耗
103         R = -r_i # 基础消耗
104         w_i_i = w_i # 基础消耗
105         if where!=s_p: # 走, 2倍
106             R = - 2 * r_i
107             w_i_i = 2 * w_i
108         if s_p == wakuang: # 挖矿钱
109             R = -3 * r_i + shouyi
110             w_i_i = 3 * w_i
111         w = w + w_i_i
112         w_chunzhuang = w_chunzhuang + w_i_i
113
114         if plan[0] >= w_i_i: # 不需要村庄
115             plan[0] = plan[0] - w_i_i
116         else: # 需要用村庄
117             plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
118             plan[0] = 0
119             if where!=s_p: # 走, 2倍

```

```

120         R = - 4 * r_i
121         if s_p == wakuang: # 挖矿钱
122             R = -6 * r_i + shouyi # 挖矿钱
123         if plan[1] < 0:
124             flag = True # 死了
125
126         print("第{}天执行{}->{}".format(t,where,s_p), plan,
              w_chunzhuang,w)
127
128         # 村庄买东西
129         if where in cunzhuang: # 村庄
130             plan[1] = plan[1] + w_chunzhuang
131             w_chunzhuang = 0
132
133         Rs.append(R)
134     return plan,Rs,flag
135
136
137 def get_path(mazes, path_map):
138     path = starts_path.copy()
139     A = starts_A.copy()
140     s = SSSSS
141     for t in range(start_t, len(weathers)-1): # (10, 30)
142         map = mazes[t,:,:]
143         if s==wakuang - 1:
144             break
145         a = probability_fun(path_map[s], map[s,:], 0)
146         s_p = a if a!=wakuang else s
147
148         path.append(s_p)
149         A.append(a)
150         s = s_p
151     return path, A
152
153

```



```

154 if __name__ == '__main__':
155     mazes = []      # 时间图Q
156     for i in range(len(weathers)):
157         mazes.append(maze)
158     mazes = np.array(mazes).copy()
159     path_map = get_pathmap(maze)
160     print(mazes.shape) # mazes (时间, 状态, 动作)
161     print(path_map)
162     print(path_map[21])
163
164     # R放入初值中
165     # for t in range(len(weathers)):
166     #     for j in range(wakuang): # 初始化所有可行域
167     #         for a in path_map[j]:
168     #             w_i, r_i = get_w_i(weathers[t])
169     #             R = -r_i
170     #             if a != j: # 走, 2倍
171     #                 R = - 2 * r_i
172     #             if a == maze.shape[1] - 1:
173     #                 R = -3 * r_i + shouyi # 挖矿钱
174     #                 mazes[t,j,a] = R
175
176     # mazes付初值
177     if ISREAD_MAZES:
178         mazes =
179             np.array(np.load("./data/{_}_{_}.npz".format(PATH,guanqia))['data'])
180
181     # 初值的可视化
182     cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True, dark=0.7,
183                                  light=0.3)
184     sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True, cmap=cmap) #
185     plt.show()
186
187     print('##### Q-learning #####')
188     planssss, Rs, flag = get_W_status(starts_path, starts_A)

```

```

187     print(flag, planssss)
188     start_t = len(starts_path)-1
189     print("起点", SSSSS, "第几天", start_t, len(weathers)-1, "矿位置",
           wakuang)
190     for k in range(learn_num):# 迭代次数
191         s = SSSSS # 起点
192
193         plan = planssss.copy() # 计划购买栈
194         path = starts_path.copy()
195         w = 1200-planssss[0] # 背包负重
196         w_chunzhuang = 1200-planssss[0] # 村庄买的东西
197         A = starts_A.copy()
198         flag = False
199         # print("*"*50)
200         # is_kuang = False
201         # print(start_t, len(weathers))
202         for t in range(start_t, len(weathers)-1):#(10, 30)
203             a = probability_fun(path_map[s], mazes[t, s, :], k=k) # 动作
204             # if s not in kuang and a in kuang:
205             #     is_kuang=True
206             # if is_kuang:
207             #     a=s
208             #     is_kuang = False
209
210             s_p = a if a != wakuang else s # 下一步要干嘛
211
212             w_i, r_i = get_w_i(weathers[t])
213             R = -r_i # 走一步消耗
214             w_i_i = w_i
215             if a != s: # 走, 2倍
216                 R = - 2 * r_i
217                 w_i_i = 2 * w_i
218             if a == wakuang:
219                 R = -3 * r_i + shouyi # 挖矿钱
220                 w_i_i = 3 * w_i

```

```

221     if weathers[t] == 3: # 沙包必须停留
222         s_p = s
223         # if a!=wakuang and a!=s:
224         #     a=s
225
226     w = w + w_i_i
227     w_chunzhuang = w_chunzhuang + w_i_i
228
229     if plan[0] >= w_i_i: # 不需要村庄
230         plan[0] = plan[0] - w_i_i
231     else: # 需要用村庄
232         plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
233         plan[0] = 0
234         if s != s_p: # 走, 2倍
235             R = - 4 * r_i
236         if a == wakuang: # 挖矿钱
237             R = -6 * r_i + shouyi # 挖矿钱
238         if plan[1] < 0:
239             flag = True # 死了
240     if plan[1] < 0:
241         flag = True # 死了
242
243     # print("第{}天执行{}, {}->{}".format(t,a,s,s_p), plan)
244
245     # 村庄买东西
246     if s in cunzhuang: # 村庄
247         plan[1] = plan[1] + w_chunzhuang
248         w_chunzhuang = 0
249
250     max_q = mazes[t + 1, s_p, probability_fun(path_map[s_p],
251         mazes[t + 1, s_p, :], 0)] # 找到最大位置的q值
252
253     if s_p == wakuang - 1: # 终点
254         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
255         s = s_p # 更新位置

```

```

255         # print(path)
256         break          # 29
257     elif flag: # 中止条件
258         R = R - 1000000
259         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
260             + gamma * max_q)
261         s = s_p # 更新位置
262         break
263     elif t == len(weathers) - 1 -1 and s_p != wakuang-1:
264         R = R - 1000000
265         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
266             + gamma * max_q)
267         s = s_p # 更新位置
268         break
269     else:
270         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
271             + gamma * max_q)
272         s = s_p # 更新位置
273
274     A.append(a)
275     path.append(s)
276     # break
277
278     np.savez_compressed("./data/{_}_{_}.npz".format(PATH,guanqia),
279         data=mazes)
280
281     path, A = get_path(mazes, path_map)
282     print(path)
283     print(A)
284     #
285     #
286     # for t in range(start_t, len(weathers)):
287     #     map = mazes[t,:,:]
288     #     for i in range(map.shape[0]):
289     #         for j in range(map.shape[1]):
290     #             if map[i,j]<-100000 and not np.isnan(map[i,j]):

```

```

286         #             map[i, j] = -1000000
287         #     cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True,
288             dark=0.7, light=0.3)
289         #     sns.heatmap(map, linewidths=0.05, cbar=True, cmap=cmap) #
290         #     plt.show()

```

A.4 第三关代码

第三关基于期望学习策略 Q-learning 学习代码，输出序列为 agent 路径 [1, 4, 6, 13]

```

1  import numpy as np
2  from maze import get_maze
3  import random
4  import matplotlib.pyplot as plt
5  import seaborn as sns
6
7  jilu_jiage = []
8
9  # ISREAD_MAZES = True
10
11  guanqia = 3 #关卡
12  learn_num = 100000 #迭代次数
13  W_UP = 1200 #负重上线
14  aleph, gamma = 0.8, 0.8 #学习率
15  shouyi = 200 # 挖矿收益
16  ddd = 10 # 天数
17  cunzhuang = []
18  kuang = [9] # 矿山
19  maze, _ = get_maze(guanqia)
20  wakuang = maze.shape[0] # 终点位置
21  # PATH = 'demo4'
22  starts_path= [1]
23  starts_A= []
24  start_t = len(starts_path) - 1
25  SSSSS = starts_path[len(starts_path) - 1]# 起点
26  shabao = 0

```

```

27
28
29 def get_weather(gaowengailv):
30     weatherssss = []
31     for i in range(ddd):
32         x = random.random()
33         if x<shabao:
34             weatherssss.append(3)
35         elif x>=shabao and x<gaowengailv:
36             weatherssss.append(2)
37         else:
38             weatherssss.append(1)
39     return weatherssss
40
41
42 def get_w_i(weather):
43     w_i, p_i = 0, 0
44     if weather==1:
45         w_i = 3*3+2*4
46         p_i = 3*5+10*4
47     elif weather==2:
48         w_i = 9*3+2*9
49         p_i = 9*5+10*9
50     else:
51         w_i = 3*10+2*10
52         p_i = 5*10+10*10
53     return w_i, p_i
54
55
56 def get_pathmap(maze):
57     path_map = []
58     # 每个状态拥有的动作
59     for i in range(maze.shape[0]): # 初始化所有可行域
60         temp = []
61         for j in range(1, maze.shape[1]):

```

```

62         if not np.isnan(maze[i,j]):
63             temp.append(j)
64         path_map.append(temp)
65     return path_map
66
67
68 def probability_fun(sss, max_a, probability=0.8, k=None):
69     '''
70     :param probability: 0为贪婪, 1为随机策略, 其他为sigema策略
71     '''
72     if probability==1:
73         return random.choice(sss)
74     elif probability==0:
75         list_a = max_a.tolist()
76         q_max = float('-inf')
77         for i in sss:
78             if list_a[i] > q_max and not np.isnan(list_a[i]):
79                 q_max=list_a[i]
80         # if probability==0:
81         #     print(q_max, list_a.index(q_max))
82         return list_a.index(q_max)
83     else:
84         if k<yuzhi:
85             probability=-(1-0.5)*k/yuzhi+1
86         else:
87             probability=0
88         if np.random.random()<probability:
89             return random.choice(sss)
90         else:
91             list_a = max_a.tolist()
92             q_max = float('-inf')
93             for i in sss:
94                 if list_a[i] > q_max and not np.isnan(list_a[i]):
95                     q_max = list_a[i]
96             # if probability==0:

```

```

97         # print(q_max, list_a.index(q_max))
98         return list_a.index(q_max)
99
100
101 def get_W_status(starts_path, starts_A, weathers):
102     s = 1 # 起点
103     w = 0 # 背包负重
104     w_chunzhuang = 0 # 村庄买的东西
105     plan = [W_UP, 0] # 计划购买栈
106     Rs = []
107     flag = False # 是否饿死
108     for t in range(len(starts_path)-1):# 第0天就开始动, 第30天没有动
109         where = starts_path[t]
110         s_p = starts_A[t] if starts_A[t] != wakuang else
            where#下一步要干嘛
111
112         w_i, r_i = get_w_i(weathers[t]) # 第t天的基础消耗
113         R = -r_i # 基础消耗
114         w_i_i = w_i # 基础消耗
115         if where!=s_p: # 走, 2倍
116             R = - 2 * r_i
117             w_i_i = 2 * w_i
118         if s_p == wakuang: # 挖矿钱
119             R = -3 * r_i + shouyi
120             w_i_i = 3 * w_i
121         w = w + w_i_i
122         w_chunzhuang = w_chunzhuang + w_i_i
123
124         if plan[0] >= w_i_i: # 不需要村庄
125             plan[0] = plan[0] - w_i_i
126         else: # 需要用村庄
127             plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
128             plan[0] = 0
129             if where!=s_p: # 走, 2倍
130                 R = - 4 * r_i

```



```

131         if s_p == wakuang: # 挖矿钱
132             R = -6 * r_i + shouyi # 挖矿钱
133             if plan[1] < 0:
134                 flag = True # 死了
135
136         print("第{}天执行{}->{}".format(t,where,s_p), plan,
              w_chunzhuang,w)
137
138         # 村庄买东西
139         if where in cunzhuang: # 村庄
140             plan[1] = plan[1] + w_chunzhuang
141             w_chunzhuang = 0
142
143         Rs.append(R)
144     return plan, Rs, flag
145
146
147 def get_path(mazes, path_map, weathers):
148     path = starts_path.copy()
149     A = starts_A.copy()
150     s = SSSSS
151     for t in range(start_t, len(weathers)-1): # (10, 30)
152         map = mazes[t,:,:]
153         if s==wakuang - 1:
154             break
155         a = probability_fun(path_map[s], map[s,:], 0)
156         s_p = a if a!=wakuang else s
157
158         path.append(s_p)
159         A.append(a)
160         s = s_p
161     return path, A
162
163
164 def get_x_y(weather):

```

```

165     if weather==1:
166         x = 3
167         y = 4
168     elif weather==2:
169         x = 9
170         y = 9
171     else:
172         x = 10
173         y = 10
174     return x, y
175
176
177 def get_R_x_ys(path,weather,sss):
178     zhuan = 0
179     for i in path:
180         if i==sss:
181             zhuan = zhuan+1000
182     xs,ys=[],[]
183     s=1
184     for t in range(1,len(path)):
185         a,b = get_x_y(weather[t-1])
186         s_p = path[t]
187         # s_p=get_index(s,a_pp)
188         A=a
189         B=b
190         if s_p != s and s_p!=sss: # 走, 2倍
191             A=2*a
192             B=2*b
193         if s_p == sss:
194             A = 3 * a
195             B = 3 * b
196         xs.append(A)
197         ys.append(B)
198         # print(s,"->>",s_p," ", A,B)
199         s = s_p

```

```

200
201 x=sum(xs)
202 y=sum(ys)
203 R = int((1200 - 2 * y)/3) * 5 + (x + y - int((1200 - 2 * y)/3)) * 10
    - zhuan
204 return -R,xs,ys,zhuan
205
206
207 def get_awser(maze,path, weather):
208     s = 1 # 起点
209     w = 0 # 背包负重
210     w_chunzhuang = 0 # 村庄买的东西
211     plan = [W_UP, 0] # 计划购买栈
212     Rs = []
213     flag = False
214     for t in range(1,len(path)):
215         w_i, r_i = get_w_i(weather[t-1])
216         R = -r_i # 走一步消耗
217         w_i_i = w_i
218         if path[t] != s: # 走, 2倍
219             R = - 2 * r_i
220             w_i_i = 2 * w_i
221         if path[t] == maze.shape[1]-1:
222             R = -3 * r_i + shouyi # 挖矿钱
223             w_i_i = 3 * w_i
224         w = w + w_i_i
225         w_chunzhuang = w_chunzhuang + w_i_i
226
227         if plan[0] >= w_i_i: # 不需要村庄
228             plan[0] = plan[0] - w_i_i
229             pass
230         else: # 需要用村庄
231             plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
232             plan[0] = 0
233             if path[t] != s: # 走, 2倍

```

```

234         R = - 4 * r_i
235         if path[t] == maze.shape[1]-1:
236             R = -6 * r_i + shouyi # 挖矿钱
237         if plan[1] < 0:
238             flag = True
239         # print(t,s,plan, w_chunzhuang,w)
240
241         # 村庄买东西
242         if s in cunzhuang: # 村庄
243             plan[1] = plan[1] + w_chunzhuang
244             w_chunzhuang = 0
245
246         s = path[t]
247         Rs.append(R)
248     return plan, Rs, flag
249
250
251 def main(gaowengailv, ISREAD_MAZES, PATH):
252     mazes = [] # 时间图Q
253     weathers = get_weather(gaowengailv)
254     for i in range(len(weathers)):
255         mazes.append(maze)
256     mazes = np.array(mazes).copy()
257     path_map = get_pathmap(maze)
258     # print(mazes.shape) # mazes (时间, 状态, 动作)
259     # print(path_map)
260     # print(path_map[21])
261
262     # mazes付初值
263     if ISREAD_MAZES:
264         mazes =
265             np.array(np.load("./data/{_}_{_}.npz".format(PATH,guanqia))['data'])
266     # 初值的可视化
267     # cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True,
268                                   dark=0.7, light=0.3)

```

```

267 #     sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True,
      cmap=cmap) #
268 #     plt.show()
269 mazes_demo = mazes.copy()
270
271 print('##### Q-learning #####')
272 planss, Rs, flag = get_W_status(starts_path, starts_A, weathers)
273 # print(flag, planss)
274 print("起点", SSSSS, "第几天", start_t, len(weathers)-1, "矿位置",
      wakuang)
275
276 huos = 0
277 jiage = []
278 for k in range(learn_num):# 迭代次数
279     s = SSSSS # 起点
280
281     plan = planss.copy() # 计划购买栈
282     path = starts_path.copy()
283     w = 1200-planss[0] # 背包负重
284     w_chunzhuang = 1200-planss[0] # 村庄买的东西
285     A = starts_A.copy()
286     flag = False
287     huo = 0
288     # print("*"*50)
289     # is_kuang = False
290     # print(start_t, len(weathers))
291     if ISREAD_MAZES:
292         mazes = mazes_demo.copy()
293         weathers = get_weather(gaowengailv).copy()
294
295     for t in range(start_t, len(weathers)-1):#(10, 30)
296
297         a = probability_fun(path_map[s], mazes[t, s, :], k=k) # 动作
298         # if s not in kuang and a in kuang:
299         #     is_kuang=True

```

```

300     # if is_kuang:
301     #     a=s
302     #     is_kuang = False
303
304     s_p = a if a != wakuang else s # 下一步要干嘛
305
306     w_i, r_i = get_w_i(weathers[t])
307     R = -r_i # 走一步消耗
308     w_i_i = w_i
309     if a != s: # 走, 2倍
310         R = - 2 * r_i
311         w_i_i = 2 * w_i
312     if a == wakuang:
313         R = -3 * r_i + shouyi # 挖矿钱
314         w_i_i = 3 * w_i
315     if weathers[t] == 3: # 沙包必须停留
316         s_p = s
317         # if a!=wakuang and a!=s:
318         #     a=s
319
320     w = w + w_i_i
321     w_chunzhuang = w_chunzhuang + w_i_i
322
323     if plan[0] >= w_i_i: # 不需要村庄
324         plan[0] = plan[0] - w_i_i
325     else: # 需要用村庄
326         plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
327         plan[0] = 0
328         if s != s_p: # 走, 2倍
329             R = - 4 * r_i
330             if a == wakuang: # 挖矿钱
331                 R = -6 * r_i + shouyi # 挖矿钱
332             if plan[1] < 0:
333                 flag = True # 死了
334         if plan[1] < 0:

```

```

335         flag = True # 死了
336
337     # print("第{}天执行{}, {}->{}".format(t,a,s,s_p), plan)
338
339     # 村庄买东西
340     if s in cunzhuang: # 村庄
341         plan[1] = plan[1] + w_chunzhuang
342         w_chunzhuang = 0
343
344     max_q = mazes[t + 1, s_p, probability_fun(path_map[s_p],
345         mazes[t + 1, s_p, :], 0)] # 找到最大位置的q值
346
347     if s_p == wakuang - 1: # 终点
348         huo = 1
349         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
350         s = s_p # 更新位置
351         # print(path)
352         break # 29
353     elif flag: # 中止条件
354         R = R - 1000000
355         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
356             + gamma * max_q)
357         s = s_p # 更新位置
358         break
359     elif t == len(weathers) - 1 - 1 and s_p != wakuang - 1:
360         R = R - 1000000
361         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
362             + gamma * max_q)
363         s = s_p # 更新位置
364         break
365     else:
366         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
367             + gamma * max_q)
368         s = s_p # 更新位置
369     A.append(a)

```

```

366         path.append(s)
367     # break
368
369
370     A.append(25)
371     pathsss = [1]
372     for i in A:
373         pathsss.append(i)
374     if huo:
375         huos = huos + 1 # 活着
376         R, xs, ys, zhuan = get_R_x_ys(pathsss, weathers, 26)
377         # print(R)
378         # print(pathsss)
379         jiage.append(R)
380
381     # if ISREAD_MAZES:
382     #     assert len(jiage)==huos
383     print("平均价格", sum(jiage)/huos, '活着概率',
384           round(huos*100/learn_num,2),"%")
385     np.savez_compressed("./data/{}_{}.npz".format(PATH,guanqia),
386                        data=mazes)
387
388     path, A = get_path(mazes, path_map, weathers)
389     # print(path)
390     # print(A)
391     paths = [1]
392     for i in A:
393         paths.append(i)
394
395     # print(len(paths))
396     plan, Rs, flag = get_awser(maze, paths, weathers)
397     print(paths)
398     # print(plan)
399     # print(Rs)
400     print("*****")

```



```

399     print(flag)
400     R, xs, ys, zhuan = get_R_x_ys(paths, weathers, 26)
401     print(R)
402
403
404 plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
405
406
407 if __name__ == '__main__':
408     gaowengailv = 0.1
409     ISREAD_MAZES = False
410     yuzhi = int(learn_num * 0.9) if not ISREAD_MAZES else 0 # 策略域
411
412     gaowengailv2 = gaowengailv + shabao
413     main(gaowengailv2, ISREAD_MAZES,
414          'demo3_gailv_1_01_{}'.format(int(gaowengailv * 100)))
415
416     ISREAD_MAZES = True
417     yuzhi = int(learn_num * 0.9) if not ISREAD_MAZES else 0 # 策略域
418     main(gaowengailv2, ISREAD_MAZES,
419          'demo3_gailv_1_01_{}'.format(int(gaowengailv * 100)))

```

A.5 第四关代码

第四关基于期望学习策略 Q-learning 学习代码，输出在沙暴概率 $q1=0.05$ 的条件下，收益和存活率随着高温概率的序列和图像，由于概率关系可能导致输出不一样，我们记录其中一次 Q 矩阵序列的数据并存储在附件中，并记录了各概率的收益和存活率。

```

1 import numpy as np
2 from maze import get_maze
3 import random
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6
7 jilu_jiage = []
8

```

```

9  # ISREAD_MAZES = True
10
11  guanqia = 4 #关卡
12  learn_num = 100000 #迭代次数
13  W_UP = 1200 #负重上线
14  aleph, gamma = 0.8, 0.8 #学习率
15  shouyi = 1000 # 挖矿收益
16  ddd = 30 # 天数
17  cunzhuang = [14]
18  kuang = [18] # 矿山
19  maze, _ = get_maze(guanqia)
20  wakuang = maze.shape[0] # 终点位置
21  # PATH = 'demo4'
22  starts_path= [1]
23  starts_A= []
24  start_t = len(starts_path) - 1
25  SSSSS = starts_path[len(starts_path) - 1]# 起点
26  # gaowengailv = 0.3 + 0.05 # 高温概率(其中0.05是沙暴概率, 固定)
27  shabao = 0.05
28
29
30  def get_weather(gaowengailv):
31      weatherssss = []
32      for i in range(ddd):
33          x = random.random()
34          if x<shabao:
35              weatherssss.append(3)
36          elif x>=shabao and x<gaowengailv:
37              weatherssss.append(2)
38          else:
39              weatherssss.append(1)
40      return weatherssss
41
42
43  def get_w_i(weather):

```

```

44     w_i, p_i = 0, 0
45     if weather==1:
46         w_i = 3*3+2*4
47         p_i = 3*5+10*4
48     elif weather==2:
49         w_i = 9*3+2*9
50         p_i = 9*5+10*9
51     else:
52         w_i = 3*10+2*10
53         p_i = 5*10+10*10
54     return w_i, p_i
55
56
57 def get_pathmap(maze):
58     path_map = []
59     # 每个状态拥有的动作
60     for i in range(maze.shape[0]): # 初始化所有可行域
61         temp = []
62         for j in range(1, maze.shape[1]):
63             if not np.isnan(maze[i,j]):
64                 temp.append(j)
65         path_map.append(temp)
66     return path_map
67
68
69 def probability_fun(sss, max_a, probability=0.8, k=None):
70     '''
71     :param probability: 0为贪婪, 1为随机策略, 其他为sigema策略
72     '''
73     if probability==1:
74         return random.choice(sss)
75     elif probability==0:
76         list_a = max_a.tolist()
77         q_max = float('-inf')
78         for i in sss:

```

```

79         if list_a[i] > q_max and not np.isnan(list_a[i]):
80             q_max=list_a[i]
81     # if probability==0:
82     #     print(q_max, list_a.index(q_max))
83     return list_a.index(q_max)
84 else:
85     if k<yuzhi:
86         probability=-(1-0.5)*k/yuzhi+1
87     else:
88         probability=0
89     if np.random.random()<probability:
90         return random.choice(sss)
91     else:
92         list_a = max_a.tolist()
93         q_max = float('-inf')
94         for i in sss:
95             if list_a[i] > q_max and not np.isnan(list_a[i]):
96                 q_max = list_a[i]
97         # if probability==0:
98         #     print(q_max, list_a.index(q_max))
99         return list_a.index(q_max)
100
101
102 def get_W_status(starts_path, starts_A, weathers):
103     s = 1 # 起点
104     w = 0 # 背包负重
105     w_chunzhuang = 0 # 村庄买的东西
106     plan = [W_UP, 0] # 计划购买栈
107     Rs = []
108     flag = False # 是否饿死
109     for t in range(len(starts_path)-1):# 第0天就开始动，第30天没有动
110         where = starts_path[t]
111         s_p = starts_A[t] if starts_A[t] != wakuang else
112         where#下一步要干嘛

```

```

113     w_i, r_i = get_w_i(weathers[t]) # 第t天的基础消耗
114     R = -r_i # 基础消耗
115     w_i_i = w_i # 基础消耗
116     if where!=s_p: # 走, 2倍
117         R = - 2 * r_i
118         w_i_i = 2 * w_i
119     if s_p == wakuang: # 挖矿钱
120         R = -3 * r_i + shouyi
121         w_i_i = 3 * w_i
122     w = w + w_i_i
123     w_chunzhuang = w_chunzhuang + w_i_i
124
125     if plan[0] >= w_i_i: # 不需要村庄
126         plan[0] = plan[0] - w_i_i
127     else: # 需要用村庄
128         plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
129         plan[0] = 0
130         if where!=s_p: # 走, 2倍
131             R = - 4 * r_i
132         if s_p == wakuang: # 挖矿钱
133             R = -6 * r_i + shouyi # 挖矿钱
134         if plan[1] < 0:
135             flag = True # 死了
136
137     print("第{}天执行{}->{}".format(t,where,s_p), plan,
          w_chunzhuang,w)
138
139     # 村庄买东西
140     if where in cunzhuang: # 村庄
141         plan[1] = plan[1] + w_chunzhuang
142         w_chunzhuang = 0
143
144     Rs.append(R)
145     return plan, Rs, flag
146

```

```

147
148 def get_path(mazes, path_map, weathers):
149     path = starts_path.copy()
150     A = starts_A.copy()
151     s = SSSSS
152     for t in range(start_t, len(weathers)-1): # (10, 30)
153         map = mazes[t,:,:]
154         if s==wakuang - 1:
155             break
156         a = probability_fun(path_map[s], map[s,:], 0)
157         s_p = a if a!=wakuang else s
158
159         path.append(s_p)
160         A.append(a)
161         s = s_p
162     return path, A
163
164
165 def get_x_y(weather):
166     if weather==1:
167         x = 3
168         y = 4
169     elif weather==2:
170         x = 9
171         y = 9
172     else:
173         x = 10
174         y = 10
175     return x, y
176
177
178 def get_R_x_ys(path,weather,sss):
179     zhuan = 0
180     for i in path:
181         if i==sss:

```

```

182         zhuan = zhuan+1000
183     xs,ys=[], []
184     s=1
185     for t in range(1,len(path)):
186         a,b = get_x_y(weather[t-1])
187         s_p = path[t]
188         # s_p=get_index(s,a_pp)
189         A=a
190         B=b
191         if s_p != s and s_p!=sss: # 走, 2倍
192             A=2*a
193             B=2*b
194         if s_p == sss:
195             A = 3 * a
196             B = 3 * b
197         xs.append(A)
198         ys.append(B)
199         # print(s,"->>",s_p," ", A,B)
200         s = s_p
201
202     x=sum(xs)
203     y=sum(ys)
204     R = int((1200 - 2 * y)/3) * 5 + (x + y - int((1200 - 2 * y)/3)) * 10
205         - zhuan
206
207     return -R,xs,ys,zhuan
208
209 def get_awser(maze,path, weather):
210     s = 1 # 起点
211     w = 0 # 背包负重
212     w_chunzhuang = 0 # 村庄买的东西
213     plan = [W_UP, 0] # 计划购买栈
214     Rs = []
215     flag = False
216     for t in range(1,len(path)):

```

```

216     w_i, r_i = get_w_i(weather[t-1])
217     R = -r_i # 走一步消耗
218     w_i_i = w_i
219     if path[t] != s: # 走, 2倍
220         R = - 2 * r_i
221         w_i_i = 2 * w_i
222     if path[t] == maze.shape[1]-1:
223         R = -3 * r_i + shouyi # 挖矿钱
224         w_i_i = 3 * w_i
225     w = w + w_i_i
226     w_chunzhuang = w_chunzhuang + w_i_i
227
228     if plan[0] >= w_i_i: # 不需要村庄
229         plan[0] = plan[0] - w_i_i
230         pass
231     else: # 需要用村庄
232         plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
233         plan[0] = 0
234         if path[t] != s: # 走, 2倍
235             R = - 4 * r_i
236             if path[t] == maze.shape[1]-1:
237                 R = -6 * r_i + shouyi # 挖矿钱
238     if plan[1] < 0:
239         flag = True
240     # print(t,s,plan, w_chunzhuang,w)
241
242     # 村庄买东西
243     if s in cunzhuang: # 村庄
244         plan[1] = plan[1] + w_chunzhuang
245         w_chunzhuang = 0
246
247     s = path[t]
248     Rs.append(R)
249     return plan, Rs, flag
250

```



```

251
252 def main(gaowengailv, ISREAD_MAZES, PATH):
253     mazes = []    # 时间图Q
254     weathers = get_weather(gaowengailv)
255     for i in range(len(weathers)):
256         mazes.append(maze)
257     mazes = np.array(mazes).copy()
258     path_map = get_pathmap(maze)
259     # print(mazes.shape) # mazes (时间, 状态, 动作)
260     # print(path_map)
261     # print(path_map[21])
262
263     # mazes付初值
264     if ISREAD_MAZES:
265         mazes =
266             np.array(np.load("./data/{_}_{_}.npz".format(PATH,guanqia))['data'])
267     # 初值的可视化
268     # cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True,
269         dark=0.7, light=0.3)
270     # sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True,
271         cmap=cmap) #
272     # plt.show()
273     mazes_demo = mazes.copy()
274
275     print('##### Q-learning #####')
276     planssss, Rs, flag = get_W_status(starts_path, starts_A, weathers)
277     # print(flag, planssss)
278     print("起点", SSSSS, "第几天", start_t, len(weathers)-1, "矿位置",
279         wakuang)
280
281     huos = 0
282     jiage = []
283     for k in range(learn_num):# 迭代次数
284         s = SSSSS # 起点

```

```

282     plan = planssss.copy() # 计划购买栈
283     path = starts_path.copy()
284     w = 1200-planssss[0] # 背包负重
285     w_chunzhuang = 1200-planssss[0] # 村庄买的东西
286     A = starts_A.copy()
287     flag = False
288     huo = 0
289     # print("*"*50)
290     # is_kuang = False
291     # print(start_t, len(weathers))
292     if ISREAD_MAZES:
293         mazes = mazes_demo.copy()
294         weathers = get_weather(gaowengailv).copy()
295
296     for t in range(start_t, len(weathers)-1):#(10, 30)
297
298         a = probability_fun(path_map[s], mazes[t, s, :], k=k) # 动作
299         # if s not in kuang and a in kuang:
300         #     is_kuang=True
301         # if is_kuang:
302         #     a=s
303         #     is_kuang = False
304
305         s_p = a if a != wakuang else s # 下一步要干嘛
306
307         w_i, r_i = get_w_i(weathers[t])
308         R = -r_i # 走一步消耗
309         w_i_i = w_i
310         if a != s: # 走, 2倍
311             R = - 2 * r_i
312             w_i_i = 2 * w_i
313         if a == wakuang:
314             R = -3 * r_i + shouyi # 挖矿钱
315             w_i_i = 3 * w_i
316         if weathers[t] == 3: # 沙包必须停留

```

```

317         s_p = s
318         # if a!=wakuang and a!=s:
319         #     a=s
320
321     w = w + w_i_i
322     w_chunzhuang = w_chunzhuang + w_i_i
323
324     if plan[0] >= w_i_i: # 不需要村庄
325         plan[0] = plan[0] - w_i_i
326     else: # 需要用村庄
327         plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
328         plan[0] = 0
329         if s != s_p: # 走, 2倍
330             R = - 4 * r_i
331         if a == wakuang: # 挖矿钱
332             R = -6 * r_i + shouyi # 挖矿钱
333         if plan[1] < 0:
334             flag = True # 死了
335     if plan[1] < 0:
336         flag = True # 死了
337
338     # print("第{}天执行{}, {}->{}".format(t,a,s,s_p), plan)
339
340     # 村庄买东西
341     if s in cunzhuang: # 村庄
342         plan[1] = plan[1] + w_chunzhuang
343         w_chunzhuang = 0
344
345     max_q = mazes[t + 1, s_p, probability_fun(path_map[s_p],
346         mazes[t + 1, s_p, :], 0)] # 找到最大位置的q值
347
348     if s_p == wakuang - 1: # 终点
349         huo = 1
350         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
351         s = s_p # 更新位置

```

```

351         # print(path)
352         break          # 29
353     elif flag: # 中止条件
354         R = R - 1000000
355         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
356             + gamma * max_q)
357         s = s_p # 更新位置
358         break
359     elif t == len(weathers) - 1 -1 and s_p != wakuang-1:
360         R = R - 1000000
361         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
362             + gamma * max_q)
363         s = s_p # 更新位置
364         break
365     else:
366         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
367             + gamma * max_q)
368         s = s_p # 更新位置
369         A.append(a)
370         path.append(s)
371         # break
372
373     A.append(25)
374     pathsss = [1]
375     for i in A:
376         pathsss.append(i)
377     if huo:
378         huos = huos + 1 # 活着
379         R, xs, ys, zhuan = get_R_x_ys(pathsss, weathers, 26)
380         # print(R)
381         # print(pathsss)
382         jiage.append(R)
383
384 if ISREAD_MAZES:

```

```

383     assert len(jiage)==huos
384     print("平均获得资金", sum(jiage)/huos, '活着概率',
          round(huos*100/learn_num,2),"%")
385 else:
386     np.savez_compressed("./data/{_}_{_}.npz".format(PATH,guanqia),
          data=mazes)
387
388     path, A = get_path(mazes, path_map, weathers)
389     # print(path)
390     # print(A)
391     paths = [1]
392     for i in A:
393         paths.append(i)
394
395     # print(len(paths))
396     plan, Rs, flag = get_awser(maze, paths, weathers)
397     print(paths)
398     # print(plan)
399     # print(Rs)
400     print("*****")
401     print(flag)
402     R, xs, ys, zhuan = get_R_x_ys(paths, weathers, 26)
403     print(R)
404
405
406 plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
407
408
409 if __name__ == '__main__':
410
411     for gaowengailv in np.arange(0,1-shabao,0.1):
412         ISREAD_MAZES = False
413         yuzhi = int(learn_num * 0.9) if not ISREAD_MAZES else 0 # 策略域
414
415         gaowengailv2 = gaowengailv+shabao

```

```

416     main(gaowengailv2, ISREAD_MAZES,
          'demo4_gailv_1_01_{}'.format(int(gaowengailv * 100)))
417
418     ISREAD_MAZES = True
419     yuzhi = int(learn_num * 0.9) if not ISREAD_MAZES else 0 # 策略域
420     main(gaowengailv2, ISREAD_MAZES,
          'demo4_gailv_1_01_{}'.format(int(gaowengailv * 100)))
421
422
423     huozhe = [ 78.76,83.31 , 74.14, 97.38 , 86.76,
424               75.12,67.75, 61.09, 41.09, 34.54]
425     jiage = [14165,11166.5,8238.2,3658.34,6177.5,
426             2303.523,1376.023,702.071,103.6772,-311.41]
427
428     huozhe2 = [63.39,33.8 , 26.38, 95.61 , 86.76,
429               45.12,37.75, 21.09, 15.88, 10.19]
430     jiage2 = [14565,11176.962,7463.11,4256.00, 7125.6496,
431              5160.709,3305.03,824.4983,315.22,-281.64]
432
433     x = np.arange(0,0.95,0.1)+shabao
434     fig, ax1 = plt.subplots()
435     ax1.plot(x, huozhe, label="保守策略时存活率", linestyle='--',
436             marker='*')
437     ax1.plot(x, huozhe2, label="非保守策略时存活率", linestyle='--',
438             marker='*')
439     ax1.plot([1], [0], linestyle=':')
440     plt.grid(linestyle='-.')
441     ax1.set_xlabel("高温概率")
442     ax1.set_ylabel("存活率/%")
443     fig.legend(loc=1, bbox_to_anchor=(1,1), bbox_transform=ax1.transAxes)
444     plt.show()
445
446     fig, ax2 = plt.subplots() # 做镜像处理
447     ax2.plot(x, jiage, label="保守策略时平均保留资金")
448     ax2.plot(x, jiage2, '-r',label="非保守策略时平均保留资金")

```

```

447 plt.grid(linestyle='-.')
448 ax2.set_ylabel("平均保留资金")
449 ax2.set_xlabel("高温概率")
450 fig.legend(loc=1, bbox_to_anchor=(1,1), bbox_transform=ax1.transAxes)
451 #
    plt.savefig(r'C:\Users\77526\Desktop\CUMCM\precode\pic\iteration.png')
452 plt.show()

```

A.6 第五关代码

第五关由于证明了双人博弈采取完全竞争策略，并且在第三关中概率不会影响最优路径，其最优路径为 [1,4,6,13] 或 [1,5,6,13]，其代码求出对应价格

```

1 import numpy as np
2 from maze import get_maze
3 import random
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 guanqia = 5 #关卡
7 learn_num = 10000 #迭代次数
8 yuzhi = int(learn_num*0.9) # 策略域
9 W_UP = 1200 #负重上线
10 aleph, gamma = 1, 0.8 #学习率
11 shouyi = 200 # 挖矿收益
12 cunzhuang = [15]
13 ISREAD_MAZES = False
14
15
16 def get_w_i(weather):
17     w_i, p_i = 0, 0
18     if weather==1:
19         w_i = 3*5+2*7
20         p_i = 5*5+10*7
21     elif weather==2:
22         w_i = 3*8+2*6
23         p_i = 5*8+10*6

```

```

24     else:
25         w_i = 3*10+2*10
26         p_i = 5*10+10*10
27     return w_i, p_i
28
29
30 def probability_fun(sss, max_a, probability=0.8, k=None):
31     '''
32     :param probability: 0为贪婪, 1为随机策略, 其他为sigema策略
33     '''
34     if probability==1:
35         return random.choice(sss)
36     elif probability==0:
37         list_a = max_a.tolist()
38         q_max = float('-inf')
39         for i in sss:
40             if list_a[i] > q_max and not np.isnan(list_a[i]):
41                 q_max=list_a[i]
42         # if probability==0:
43         #     print(q_max, list_a.index(q_max))
44         return list_a.index(q_max)
45     else:
46         if k<yuzhi:
47             probability=-(1-0.5)*k/yuzhi+1
48         else:
49             probability=0
50         if np.random.random()<probability:
51             return random.choice(sss)
52         else:
53             list_a = max_a.tolist()
54             q_max = float('-inf')
55             for i in sss:
56                 if list_a[i] > q_max and not np.isnan(list_a[i]):
57                     q_max = list_a[i]
58             # if probability==0:

```



```

59         # print(q_max, list_a.index(q_max))
60         return list_a.index(q_max)
61
62
63 def get_index(s,a,map):
64     if a!=map.shape[1]-1:
65         return a
66     else:
67         return s
68
69
70 def get_path(mazes, weather, path_map):
71     s=1
72     path = []
73     path.append(s)
74     for t in range(len(weather)):
75         map = mazes[t,:,:]
76         if s==map.shape[0]-1:
77             break
78         a = probability_fun(path_map[s], map[s,:], 0)
79         path.append(a)
80         s_p = get_index(s, a, map)
81         s = s_p
82     return path
83
84
85 def get_pathmap(maze):
86     path_map = []
87     #每个状态拥有的动作
88     for i in range(maze.shape[0]): # 初始化所有可行域
89         temp = []
90         for j in range(1, maze.shape[1]):
91             if not np.isnan(maze[i,j]):
92                 temp.append(j)
93     path_map.append(temp)

```

```

94     return path_map
95
96
97 def init_mazes(mazes, path, weather):
98     for k in range(10000):
99         # 判断终止条件
100         print("*" * 50)
101         s = 1 # 起点
102         w = 0 # 背包负重
103         w_chunzhuang = 0 # 村庄买的东西
104         plan = [W_UP, 0] # 计划购买栈
105         flag = False
106
107         for t in range(1, len(path)):
108
109             a = path[t]
110             s_p=get_index(s, a, mazes[t,:,:]) # 动作执行完后状态
111
112             w_i, r_i = get_w_i(weather[t])
113             R = -r_i # 走一步消耗
114             w_i_i = w_i
115             if a != s: # 走, 2倍
116                 R = - 2 * r_i
117                 w_i_i = 2 * w_i
118             if a == maze.shape[1] - 1:
119                 R = -3 * r_i + shouyi # 挖矿钱
120                 w_i_i = 3 * w_i
121
122             w = w + w_i_i
123             w_chunzhuang = w_chunzhuang + w_i_i
124
125             #####
126             # 村庄买东西
127             if plan[0] >= w_i_i: # 不需要村庄
128                 plan[0] = plan[0] - w_i_i

```

```

129         pass
130     else: # 需要用村庄
131         plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
132         plan[0] = 0
133         if path[t] != s: # 走, 2倍
134             R = - 4 * r_i
135         if path[t] == maze.shape[1] - 1:
136             R = -6 * r_i + shouyi # 挖矿钱
137         if plan[1] < 0:
138             flag = True
139
140     # 村庄买东西
141     if s in cunzhuang: # 村庄
142         plan[1] = plan[1] + w_chunzhuang
143         w_chunzhuang = 0
144     #####
145     max_q = mazes[t + 1, s_p, probability_fun(path_map[s_p],
146         mazes[t + 1, s_p, :], 0)] # 找到最大位置的q值
147     if 15 in path:
148         print(path, max_q, plan, w, w_chunzhuang, R, )
149
150     if s_p == lenth - 1: # 终点
151         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
152         s = s_p # 更新位置
153         break
154     elif flag or (t == len(weather) - 1 - 1 and s_p != lenth -
155         1): # 中止条件
156         R = R - 1000000
157         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
158             + gamma * max_q)
159         s = s_p # 更新位置
160         break
161     else:
162         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
163             + gamma * max_q)

```

```

160         s = s_p # 更新位置
161     np.savez_compressed("./data/guan_init{}.npz".format(guanqia),
162                        data=mazes)
163
164     return mazes
165
166 def get_awser(maze,path, weather):
167     s = 1 # 起点
168     w = 0 # 背包负重
169     w_chunzhuang = 0 # 村庄买的东西
170     plan = [W_UP, 0] # 计划购买栈
171     Rs = []
172     flag = False
173     for t in range(1,len(path)):
174         w_i, r_i = get_w_i(weather[t-1])
175         R = -r_i # 走一步消耗
176         w_i_i = w_i
177         if path[t] != s: # 走, 2倍
178             R = - 2 * r_i
179             w_i_i = 2 * w_i
180         if path[t] == maze.shape[1]-1:
181             R = -3 * r_i + shouyi # 挖矿钱
182             w_i_i = 3 * w_i
183         w = w + w_i_i
184         w_chunzhuang = w_chunzhuang + w_i_i
185
186         if plan[0] >= w_i_i: # 不需要村庄
187             plan[0] = plan[0] - w_i_i
188             pass
189         else: # 需要用村庄
190             plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
191             plan[0] = 0
192             if path[t] != s: # 走, 2倍
193                 R = - 4 * r_i
194             if path[t] == maze.shape[1]-1:

```

```

194         R = -6 * r_i + shouyi # 挖矿钱
195     if plan[1] < 0:
196         flag = True
197     print(t,s,plan, w_chunzhuang,w)
198
199     # 村庄买东西
200     if s in cunzhuang: # 村庄
201         plan[1] = plan[1] + w_chunzhuang
202         w_chunzhuang = 0
203
204     s = path[t]
205     Rs.append(R)
206     return plan,Rs,flag
207
208
209 from get_R import get_R_x_y
210
211 if __name__ == '__main__':
212     maze, weather = get_maze(guanqia)
213     path = [1,4,6,13]
214     print(len(path))
215     plan,Rs, flag = get_awser(maze, path, weather)
216     print(path)
217     print(plan)
218     print(Rs)
219     print("*****")
220     print(flag)
221     R,xs,ys = get_R_x_y(path, weather, 28)
222     print(R)
223
224 if __name__ == '__main__ssss;':
225     maze, weather = get_maze(guanqia)
226     lenth = maze.shape[0] #终点位置
227     print(weather, lenth)
228     mazes = [] # 时间图

```

```

229     for i in range(len(weather)):
230         mazes.append(maze)
231     mazes = np.array(mazes)
232     path_map = get_pathmap(maze)
233     print(mazes.shape) # mazes (时间, 状态, 动作)
234     print(path_map)
235
236     # R放入初值中
237     for t in range(len(weather)):
238         for j in range(lenth): # 初始化所有可行域
239             for a in path_map[j]:
240                 w_i, r_i = get_w_i(weather[t])
241                 R = -r_i
242                 if a != j: # 走, 2倍
243                     R = - 2 * r_i
244                 if a == maze.shape[1] - 1:
245                     R = -3 * r_i + shouyi # 挖矿钱
246                 mazes[t,j,a] = R
247
248     init_path = [1, 25, 24, 23, 23, 22, 9, 9, 15, 14, 12, 28, 28, 28,
249                 28, 28, 12, 28, 28, 13, 15, 9, 21, 27]
250
251     cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True, dark=0.7,
252                                  light=0.3)
253
254     sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True, cmap=cmap) #
255     plt.show()
256
257     mazes = init_mazes(mazes,init_path,weather)
258     print(mazes)
259     # mazes =
260         np.array(np.load("./data/guan_init{}.npz".format(guanqia))['data'])
261
262     if ISREAD_MAZES:
263         mazes =
264             np.array(np.load("./data/guan{}.npz".format(guanqia))['data'])

```

```

260
261 cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True, dark=0.7,
    light=0.3)
262 sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True, cmap=cmap) #
263 plt.show()
264
265
266 ##q-learning
267 print('##### Q-learning #####')
268 for k in range(learn_num):# 迭代次数
269     # 判断终止条件
270     print("*"*50)
271     s = 1 # 起点
272     w = 0 # 背包负重
273     w_chunzhuang = 0 # 村庄买的东西
274     plan = [W_UP, 0] # 计划购买栈
275     path = [s]
276     flag = False
277
278     for t in range(len(weather)-1):
279         a = probability_fun(path_map[s], mazes[t,s,:], k=k) #动作
280         s_p = get_index(s, a, mazes[t,:,:]) # 动作执行完后状态
281         w_i, r_i = get_w_i(weather[t])
282         R = -r_i # 走一步消耗
283         w_i_i = w_i
284         if a != s: # 走, 2倍
285             R = - 2 * r_i
286             w_i_i = 2*w_i
287         if a==maze.shape[1]-1:
288             R = -3*r_i+shouyi # 挖矿钱
289             w_i_i = 3*w_i
290         if weather[t]==3: # 沙包必须停留
291             s_p = s
292
293         w = w + w_i_i

```

```

294     w_chunzhuang = w_chunzhuang + w_i_i
295
296     #####
297     # 村庄买东西
298     if plan[0] >= w_i_i: # 不需要村庄
299         plan[0] = plan[0] - w_i_i
300         pass
301     else: # 需要用村庄
302         plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
303         plan[0] = 0
304         if path[t] != s: # 走, 2倍
305             R = - 4 * r_i
306         if path[t] == maze.shape[1] - 1:
307             R = -6 * r_i + shouyi # 挖矿钱
308         if plan[1] < 0:
309             flag = True
310
311     # 村庄买东西
312     if s in cunzhuang: # 村庄
313         plan[1] = plan[1] + w_chunzhuang
314         w_chunzhuang = 0
315     #####
316
317     max_q = mazes[t+1, s_p, probability_fun(path_map[s_p],
318         mazes[t+1,s_p,:], 0)] # 找到最大位置的q值
319     if 15 in path:
320         print(path, max_q, plan, w, w_chunzhuang, R,)
321
322     if s_p==lenth-1:# 终点
323         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
324         s = s_p # 更新位置
325         break
326     elif flag or (t==len(weather)-1-1 and s_p != lenth-1):#中止条件
327         R = R - 1000000
328         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R

```



```

        + gamma*max_q)
328     s = s_p # 更新位置
329     break
330 else:
331     mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
        + gamma*max_q)
332     s = s_p # 更新位置
333     # print(mazes[t,s,a])
334     path.append(a)
335     pass
336
337
338 # 输出路径
339 # print(mazes)
340 np.savez_compressed("./data/guan{}.npz".format(guanqia), data=mazes)
341
342 path = get_path(mazes, weather, path_map)
343 print(path)
344
345 #
346 # for t in range(25, len(weather)):
347 #     map = mazes[t,:,:]
348 #     for i in range(map.shape[0]):
349 #         for j in range(map.shape[1]):
350 #             if map[i,j]<-100000 and not np.isnan(map[i,j]):
351 #                 map[i, j] = -1000000
352 #     cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True,
        dark=0.7, light=0.3)
353 #     sns.heatmap(map, linewidths=0.05, cbar=True, cmap=cmap) #
354 #     plt.show()

```

A.7 第六关代码

第六关基于期望学习策略的三 agent 完全合作 Q-learning 学习代码，输出于第四关类似，图像分别表示三个 agents 的存活率和收益曲线。

```

1 import numpy as np
2 from maze import get_maze
3 import random
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6
7 jilu_jiage = []
8
9 # ISREAD_MAZES = True
10
11 guanqia = 6 #关卡
12 learn_num = 100000 #迭代次数
13 W_UP = 1200 #负重上线
14 aleph, gamma = 0.8, 0.8 #学习率
15 shouyi = 1000 # 挖矿收益
16 ddd = 30 # 天数
17 cunzhuang = [14]
18 kuang = [18] # 矿山
19 maze, _ = get_maze(guanqia)
20 wakuang = maze.shape[0] # 终点位置
21 # PATH = 'demo4'
22 starts_path= [1]
23 starts_A= []
24 start_t = len(starts_path) - 1
25 SSSSS = starts_path[len(starts_path) - 1]# 起点
26 # gaowengailv = 0.3 + 0.05 # 高温概率(其中0.05是沙暴概率, 固定)
27 shabao = 0.05
28
29
30 def get_weather(gaowengailv):
31     weatherssss = []
32     for i in range(ddd):
33         x = random.random()
34         if x<shabao:

```

```

35         weatherssss.append(3)
36     elif x>=shabao and x<gaowengailv:
37         weatherssss.append(2)
38     else:
39         weatherssss.append(1)
40     return weatherssss
41
42
43 def get_w_i(weather):
44     w_i, p_i = 0, 0
45     if weather==1:
46         w_i = 3*3+2*4
47         p_i = 3*5+10*4
48     elif weather==2:
49         w_i = 9*3+2*9
50         p_i = 9*5+10*9
51     else:
52         w_i = 3*10+2*10
53         p_i = 5*10+10*10
54     return w_i, p_i
55
56
57 def get_pathmap(maze):
58     path_map = []
59     # 每个状态拥有的动作
60     for i in range(maze.shape[0]): # 初始化所有可行域
61         temp = []
62         for j in range(1, maze.shape[1]):
63             if not np.isnan(maze[i,j]):
64                 temp.append(j)
65         path_map.append(temp)
66     return path_map
67
68
69 def probability_fun(sss, max_a, probability=0.8, k=None):

```

```

70     '''
71     :param probability: 0为贪婪, 1为随机策略, 其他为sigema策略
72     '''
73     if probability==1:
74         return random.choice(sss)
75     elif probability==0:
76         list_a = max_a.tolist()
77         q_max = float('-inf')
78         for i in sss:
79             if list_a[i] > q_max and not np.isnan(list_a[i]):
80                 q_max=list_a[i]
81         # if probability==0:
82         #     print(q_max, list_a.index(q_max))
83         return list_a.index(q_max)
84     else:
85         if k<yuzhi:
86             probability=-(1-0.5)*k/yuzhi+1
87         else:
88             probability=0
89         if np.random.random()<probability:
90             return random.choice(sss)
91         else:
92             list_a = max_a.tolist()
93             q_max = float('-inf')
94             for i in sss:
95                 if list_a[i] > q_max and not np.isnan(list_a[i]):
96                     q_max = list_a[i]
97             # if probability==0:
98             #     print(q_max, list_a.index(q_max))
99             return list_a.index(q_max)
100
101
102 def get_W_status(starts_path, starts_A, weathers):
103     s = 1 # 起点
104     w = 0 # 背包负重

```

```

105 w_chunzhuang = 0 # 村庄买的东西
106 plan = [W_UP, 0] # 计划购买栈
107 Rs = []
108 flag = False # 是否饿死
109 for t in range(len(starts_path)-1):# 第0天就开始动，第30天没有动
110     where = starts_path[t]
111     s_p = starts_A[t] if starts_A[t] != wakuang else
        where#下一步要干嘛
112
113     w_i, r_i = get_w_i(weathers[t]) # 第t天的基础消耗
114     R = -r_i # 基础消耗
115     w_i_i = w_i # 基础消耗
116     if where!=s_p: # 走，2倍
117         R = - 2 * r_i
118         w_i_i = 2 * w_i
119     if s_p == wakuang: # 挖矿钱
120         R = -3 * r_i + shouyi
121         w_i_i = 3 * w_i
122     w = w + w_i_i
123     w_chunzhuang = w_chunzhuang + w_i_i
124
125     if plan[0] >= w_i_i: # 不需要村庄
126         plan[0] = plan[0] - w_i_i
127     else: # 需要用村庄
128         plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
129         plan[0] = 0
130         if where!=s_p: # 走，2倍
131             R = - 4 * r_i
132         if s_p == wakuang: # 挖矿钱
133             R = -6 * r_i + shouyi # 挖矿钱
134         if plan[1] < 0:
135             flag = True # 死了
136
137     print("第{}天执行{}->{}".format(t,where,s_p), plan,
        w_chunzhuang,w)

```

```

138
139     # 村庄买东西
140     if where in cunzhuang: # 村庄
141         plan[1] = plan[1] + w_chunzhuang
142         w_chunzhuang = 0
143
144     Rs.append(R)
145     return plan, Rs, flag
146
147
148 def get_path(mazes, path_map, weathers):
149     path = starts_path.copy()
150     A = starts_A.copy()
151     s = SSSSS
152     for t in range(start_t, len(weathers)-1): # (10, 30)
153         map = mazes[t,:,:]
154         if s==wakuang - 1:
155             break
156         a = probability_fun(path_map[s], map[s,:], 0)
157         s_p = a if a!=wakuang else s
158
159         path.append(s_p)
160         A.append(a)
161         s = s_p
162     return path, A
163
164
165 def get_x_y(weather):
166     if weather==1:
167         x = 3
168         y = 4
169     elif weather==2:
170         x = 9
171         y = 9
172     else:

```

```

173     x = 10
174     y = 10
175     return x, y
176
177
178 def get_R_x_ys(path,weather,sss):
179     zhuan = 0
180     for i in path:
181         if i==sss:
182             zhuan = zhuan+1000
183     xs,ys=[],[]
184     s=1
185     for t in range(1,len(path)):
186         a,b = get_x_y(weather[t-1])
187         s_p = path[t]
188         # s_p=get_index(s,a_pp)
189         A=a
190         B=b
191         if s_p != s and s_p!=sss: # 走, 2倍
192             A=2*a
193             B=2*b
194         if s_p == sss:
195             A = 3 * a
196             B = 3 * b
197         xs.append(A)
198         ys.append(B)
199         # print(s,"->>",s_p," ", A,B)
200         s = s_p
201
202     x=sum(xs)
203     y=sum(ys)
204     R = int((1200 - 2 * y)/3) * 5 + (x + y - int((1200 - 2 * y)/3)) * 10
205         - zhuan
206     return -R,xs,ys,zhuan

```

```

207
208 def get_awser(maze,path, weather):
209     s = 1 # 起点
210     w = 0 # 背包负重
211     w_chunzhuang = 0 # 村庄买的东西
212     plan = [W_UP, 0] # 计划购买栈
213     Rs = []
214     flag = False
215     for t in range(1,len(path)):
216         w_i, r_i = get_w_i(weather[t-1])
217         R = -r_i # 走一步消耗
218         w_i_i = w_i
219         if path[t] != s: # 走, 2倍
220             R = - 2 * r_i
221             w_i_i = 2 * w_i
222         if path[t] == maze.shape[1]-1:
223             R = -3 * r_i + shouyi # 挖矿钱
224             w_i_i = 3 * w_i
225         w = w + w_i_i
226         w_chunzhuang = w_chunzhuang + w_i_i
227
228         if plan[0] >= w_i_i: # 不需要村庄
229             plan[0] = plan[0] - w_i_i
230             pass
231         else: # 需要用村庄
232             plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
233             plan[0] = 0
234             if path[t] != s: # 走, 2倍
235                 R = - 4 * r_i
236                 if path[t] == maze.shape[1]-1:
237                     R = -6 * r_i + shouyi # 挖矿钱
238             if plan[1] < 0:
239                 flag = True
240             # print(t,s,plan, w_chunzhuang,w)
241

```



```

242     # 村庄买东西
243     if s in cunzhuang: # 村庄
244         plan[1] = plan[1] + w_chunzhuang
245         w_chunzhuang = 0
246
247     s = path[t]
248     Rs.append(R)
249     return plan, Rs, flag
250
251
252 def main(gaowengailv, ISREAD_MAZES, PATH):
253     mazes = [] # 时间图Q
254     weathers = get_weather(gaowengailv)
255     for i in range(len(weathers)):
256         mazes.append(maze)
257     mazes = np.array(mazes).copy()
258     path_map = get_pathmap(maze)
259     # print(mazes.shape) # mazes (时间, 状态, 动作)
260     # print(path_map)
261     # print(path_map[21])
262
263     # mazes付初值
264     if ISREAD_MAZES:
265         mazes =
266             np.array(np.load("./data/{ }_{ }.npz".format(PATH, guanqia))['data'])
267     # 初值的可视化
268     # cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True,
269     #                               dark=0.7, light=0.3)
270     # sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True,
271     #              cmap=cmap) #
272     # plt.show()
273     mazes_demo = mazes.copy()
274
275     print('##### Q-learning #####')
276     planss, Rs, flag = get_W_status(starts_path, starts_A, weathers)

```

```

274 # print(flag, planssss)
275 print("起点", SSSSS, "第几天", start_t, len(weathers)-1, "矿位置",
      wakuang)
276
277 huos = 0
278 jiage = []
279 for k in range(learn_num):# 迭代次数
280     s = SSSSS # 起点
281
282     plan = planssss.copy() # 计划购买栈
283     path = starts_path.copy()
284     w = 1200-planssss[0] # 背包负重
285     w_chunzhuang = 1200-planssss[0] # 村庄买的东西
286     A = starts_A.copy()
287     flag = False
288     huo = 0
289     # print("*"*50)
290     # is_kuang = False
291     # print(start_t, len(weathers))
292     if ISREAD_MAZES:
293         mazes = mazes_demo.copy()
294         weathers = get_weather(gaowengailv).copy()
295
296     for t in range(start_t, len(weathers)-1):#(10, 30)
297
298         a = probability_fun(path_map[s], mazes[t, s, :], k=k) # 动作
299         # if s not in kuang and a in kuang:
300             #     is_kuang=True
301         # if is_kuang:
302             #     a=s
303         #     is_kuang = False
304
305         s_p = a if a != wakuang else s # 下一步要干嘛
306
307         w_i, r_i = get_w_i(weathers[t])

```

```

308     R = -r_i # 走一步消耗
309     w_i_i = w_i
310     if a != s: # 走, 2倍
311         R = - 2 * r_i
312         w_i_i = 2 * w_i
313     if a == wakuang:
314         R = -3 * r_i + shouyi # 挖矿钱
315         w_i_i = 3 * w_i
316     if weathers[t] == 3: # 沙包必须停留
317         s_p = s
318         # if a!=wakuang and a!=s:
319         #     a=s
320
321     w = w + w_i_i
322     w_chunzhuang = w_chunzhuang + w_i_i
323
324     if plan[0] >= w_i_i: # 不需要村庄
325         plan[0] = plan[0] - w_i_i
326     else: # 需要用村庄
327         plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
328         plan[0] = 0
329         if s != s_p: # 走, 2倍
330             R = - 4 * r_i
331             if a == wakuang: # 挖矿钱
332                 R = -6 * r_i + shouyi # 挖矿钱
333             if plan[1] < 0:
334                 flag = True # 死了
335         if plan[1] < 0:
336             flag = True # 死了
337
338     # print("第{}天执行{}, {}->{}".format(t,a,s,s_p), plan)
339
340     # 村庄买东西
341     if s in cunzhuang: # 村庄
342         plan[1] = plan[1] + w_chunzhuang

```

```

343         w_chunzhuang = 0
344
345     max_q = mazes[t + 1, s_p, probability_fun(path_map[s_p],
346         mazes[t + 1, s_p, :], 0)] # 找到最大位置的q值
347
348     if s_p == wakuang - 1: # 终点
349         huo = 1
350         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
351         s = s_p # 更新位置
352         # print(path)
353         break # 29
354     elif flag: # 中止条件
355         R = R - 1000000
356         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
357             + gamma * max_q)
358         s = s_p # 更新位置
359         break
360     elif t == len(weathers) - 1 - 1 and s_p != wakuang - 1:
361         R = R - 1000000
362         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
363             + gamma * max_q)
364         s = s_p # 更新位置
365         break
366     else:
367         mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
368             + gamma * max_q)
369         s = s_p # 更新位置
370         A.append(a)
371         path.append(s)
372         # break
373
374     A.append(25)
375     pathsss = [1]
376     for i in A:

```

```

374         pathssss.append(i)
375     if huo:
376         huos = huos + 1 # 活着
377         R, xs, ys, zhuan = get_R_x_ys(pathssss, weathers, 26)
378         # print(R)
379         # print(pathssss)
380         jiage.append(R)
381
382     if ISREAD_MAZES:
383         assert len(jiage)==huos
384         print("平均收益", sum(jiage)/huos, '活着概率',
385               round(huos*100/learn_num,2),"%")
386     else:
387         np.savez_compressed("./data/{_}_{_}.npz".format(PATH,guanqia),
388                             data=mazes)
389
390     path, A = get_path(mazes, path_map, weathers)
391     # print(path)
392     # print(A)
393     paths = [1]
394     for i in A:
395         paths.append(i)
396
397     # print(len(paths))
398     plan, Rs, flag = get_awser(maze, paths, weathers)
399     print(paths)
400     # print(plan)
401     # print(Rs)
402     print("*****")
403     print(flag)
404     R, xs, ys, zhuan = get_R_x_ys(paths, weathers, 26)
405     print(R)
406
407 plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签

```

```

407
408
409 if __name__ == '__main__':
410
411     # for gaowengailv in np.arange(0,1-shabao,0.1):
412     #     ISREAD_MAZES = False
413     #     yuzhi = int(learn_num * 0.9) if not ISREAD_MAZES else 0 # 策略域
414     #
415     #     gaowengailv2 = gaowengailv+shabao
416     #     main(gaowengailv2, ISREAD_MAZES,
417               'demo6_gailv_2_01_{}'.format(int(gaowengailv * 100)))
418     #
419     #     ISREAD_MAZES = True
420     #     yuzhi = int(learn_num * 0.9) if not ISREAD_MAZES else 0 # 策略域
421     #     main(gaowengailv2, ISREAD_MAZES,
422               'demo6_gailv_2_01_{}'.format(int(gaowengailv * 100)))
423
424     huozhe = np.array([88.05,61.2,25.03,64.89,34.8,
425                       30.23,80.05 ,12.1 ,21.5, 87.07])
426     jiage = np.array([10461.8539,13351.1500,9176.8173,5457.4128,5252.251,
427                      4217.9555, 1592.151,3037.4466,399.0921,111.672])/3
428
429     huozhe2 = np.array([64.15,25.89,4.66,44.96,57.35 ,
430                        29.9, 35.93,38.49,96.08, 14.27,])
431     jiage2 =np.array( [12210.633,8490.3822,11078.8149, 6299.4749,
432                       4370.22,
433                       4201.05424,3225.40552,2148.689,-208.6307,407.588972])/3
434
435     huozhe3 = np.array([26.29 , 91.12 ,24.6,8.31,6.87 ,
436                        88.73, 35.9,38.0,50.37, 87.23,])
437     jiage3 = np.array([14165.0 ,2747.553, 9173.948, 6299.4749, 7124.87,
438                       2045.927, 3196.399,2165.566, 1098.84858,100.521])/3

```

```

439 x = np.arange(0,0.95,0.1)+shabao
440 plt.plot(x, huozhe, label="平均存活率", linestyle='--', marker='*')
441 plt.ylim((0,100))
442 plt.show()
443
444
445
446 x = np.arange(0,0.95,0.1)+shabao
447 fig, ax1 = plt.subplots()
448 ax1.plot(x, huozhe, label="agent1存活率", linestyle='--', marker='*')
449 ax1.plot(x, huozhe2, label="agent2存活率", linestyle='--', marker='*')
450 ax1.plot(x, huozhe3, label="agent3存活率", linestyle='--', marker='*')
451 ax1.plot([1], [0], linestyle=':')
452 plt.grid(linestyle='-.')
453 ax1.set_xlabel("高温概率")
454 ax1.set_ylabel("存活率/%")
455 fig.legend(loc=1, bbox_to_anchor=(1,1), bbox_transform=ax1.transAxes)
456 plt.show()
457
458 fig, ax2 = plt.subplots() # 做镜像处理
459 ax2.plot(x, jiage, label="agent1平均保留资金")
460 ax2.plot(x, jiage2, '-r', label="agent2平均保留资金")
461 ax2.plot(x, jiage3, '-k', label="agent3平均保留资金")
462 plt.grid(linestyle='-.')
463 ax2.set_ylabel("平均保留资金")
464 ax2.set_xlabel("高温概率")
465 fig.legend(loc=1, bbox_to_anchor=(1,1), bbox_transform=ax1.transAxes)
466 #
467     plt.savefig(r'C:\Users\77526\Desktop\CUMCM\precode\pic\iteration.png')
468 plt.show()

```