基于多策略 Q-learing 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-learing 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, 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, 55, 62, 55, 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-learing** 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
$\bigotimes$	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：

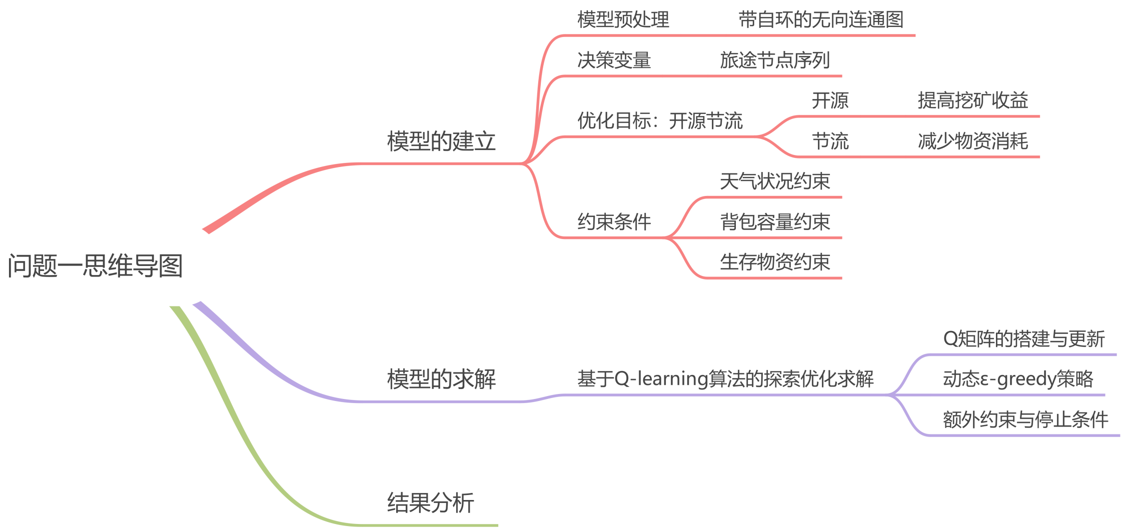
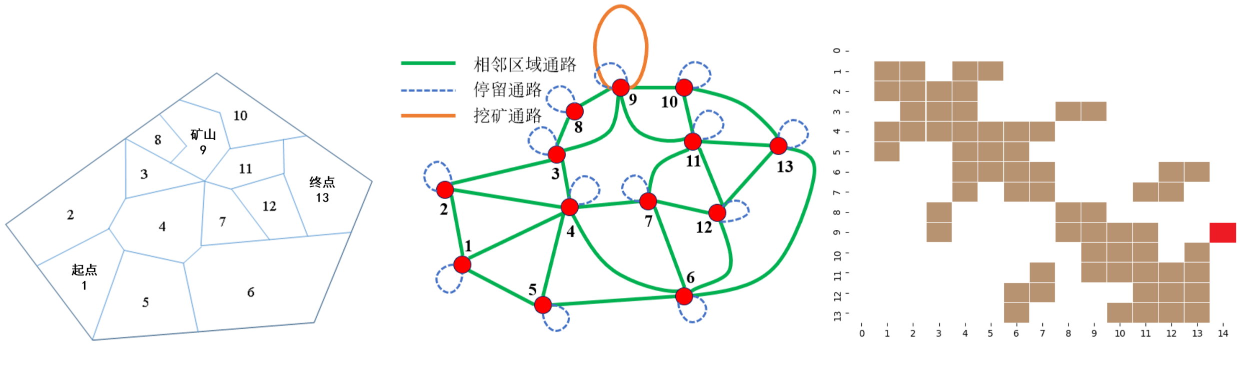


图 **2**　问题一思维流程图

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 $G(V,E,\boldsymbol{W})$。设连通图阶数为 n，则每个区域为节点 $v_i \in V(i=1,2,...,n)$，相邻区域间通道可以表示为边 $e_j \in E(j=1,2,....,m)$，权重矩阵为 $\boldsymbol{W}$，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



(a) 初始关卡地图　　(b) 关卡连通图　　(c) 连通图矩阵

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走路径。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 $s_t(t=1,2,...,n)$，采用动作为 $a_t(t=1,2,...,n)$。记截止时间为 T，则决策变量可以表示为

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

$$1\leq n\leq T, \tag{2}$$

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

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

定义符号 $\bigotimes$ 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

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

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

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

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

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

第 t 天的饮水于物资消耗可表示为

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

$\Delta w(weather_t)$ 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

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

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

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

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

$$\max_{(S_{n+1}|A_n)} M_0 + \sum_{t=1}^{n} R(s_t, a_t), \tag{6}$$

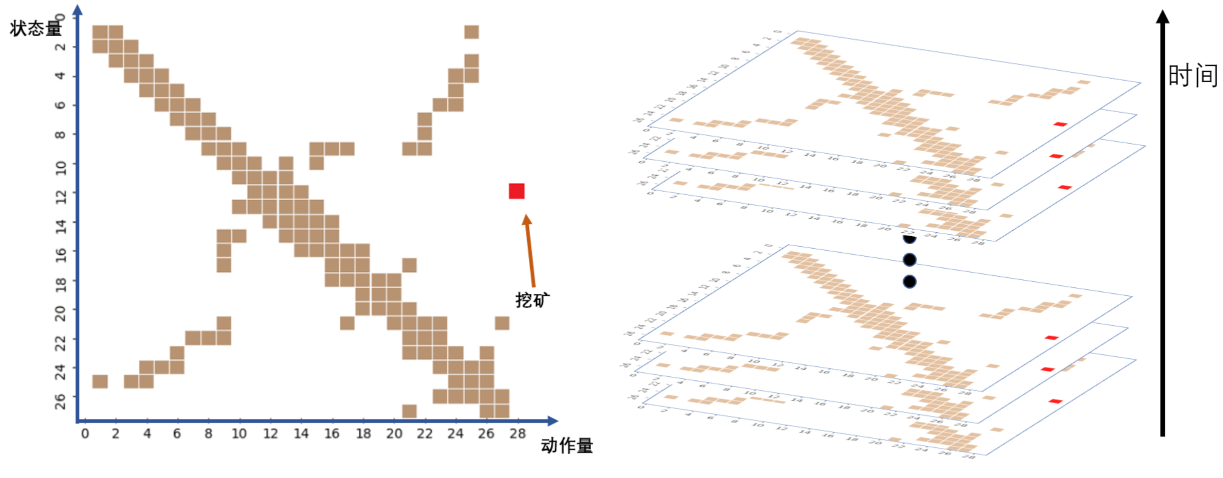
$$s.t. \quad \begin{cases} 1 \leq n \leq T, \\ s_{t+1} = s_t \bigotimes a_t, \\ s_1 = s_{outset}, \\ s_{n+1} = s_{destination}, \\ |a_t| \leqslant \frac{weather_t}{2}, \\ \forall t = 0, 1, 2, ..., T : w(t) \geqslant 0, \\ \forall t = 0, 1, 2, ..., T : f(t) \geqslant 0, \\ \forall t = 0, 1, 2, ..., T : w(t) + f(t) \leq b_0, \\ w(t+1) = w(t) - \Delta w(weather_t) \cdot \lambda(s_t, a_t), \\ f(t+1) = f(t) - \Delta f(weather_t) \cdot \lambda(s_t, a_t), \\ R(s_t, a_t, weather_t) = -consume(weather_t) \cdot \lambda(s_t, a_t) + income(s_t, a_t). \end{cases} \tag{7}$$

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



(a) 单层连通图矩阵

(b) 包含时间维度的 Q 矩阵

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 $k \leqslant \xi$ 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

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

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 Ballman 优化方程[1, 3] 表示为

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

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

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

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

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

$\lambda \in [0,1]$ 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

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

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

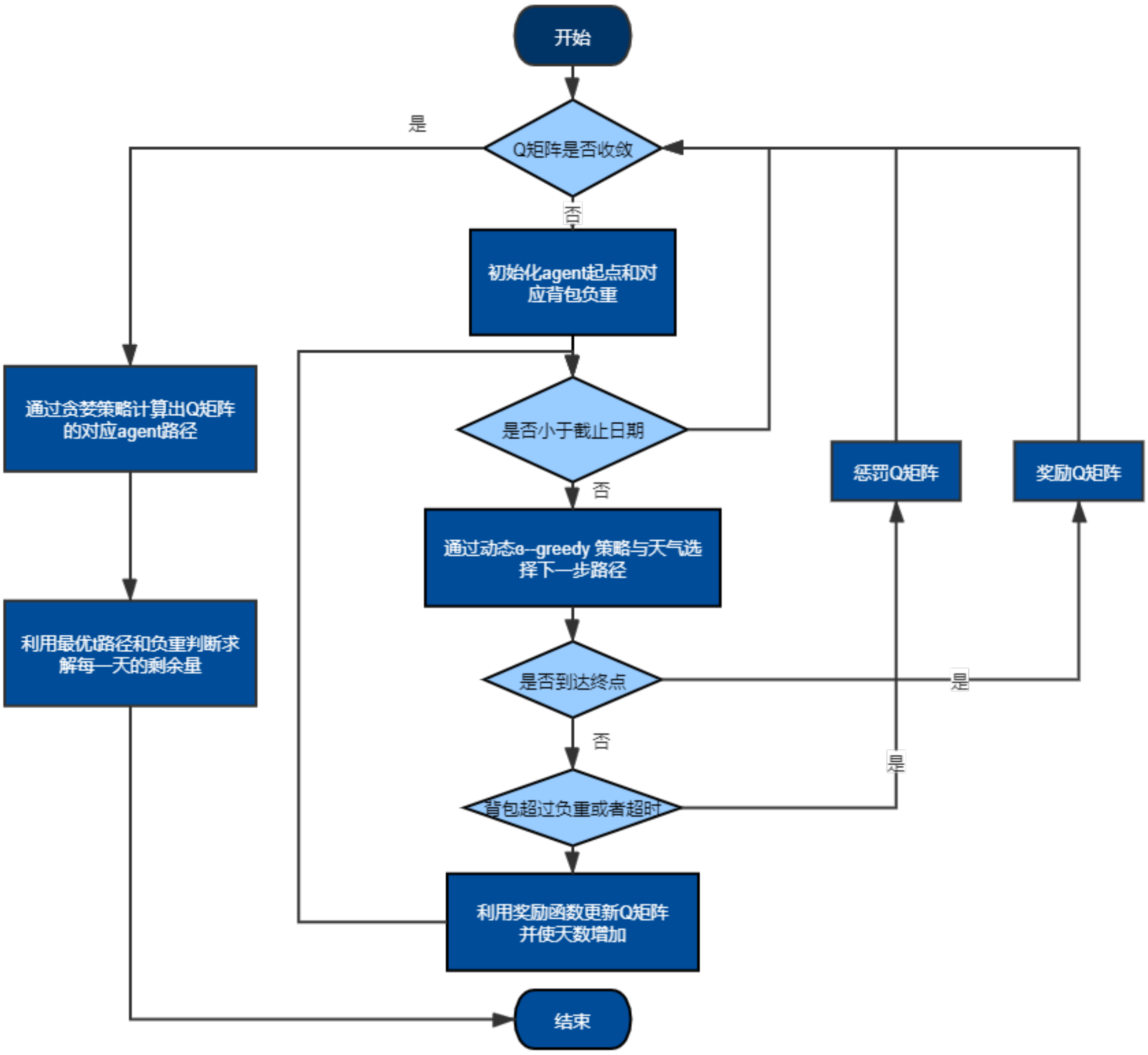

开始
Q矩阵是否收敛
是
否
初始化agent起点和对应背包负重
通过贪婪策略计算出Q矩阵的对应agent路径
是否小于截止日期
否
惩罚Q矩阵
奖励Q矩阵
通过动态ε--greedy 策略与天气选择下一步路径
利用最优路径和负重判断求解每一天的剩余量
是否到达终点
是
否
是
背包超过负重或者超时
利用奖励函数更新Q矩阵并使天数增加
结束

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-learing 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所得出第一关的最优结果如表 2 所示，该路径求下出最大收益资金为 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-learing 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：

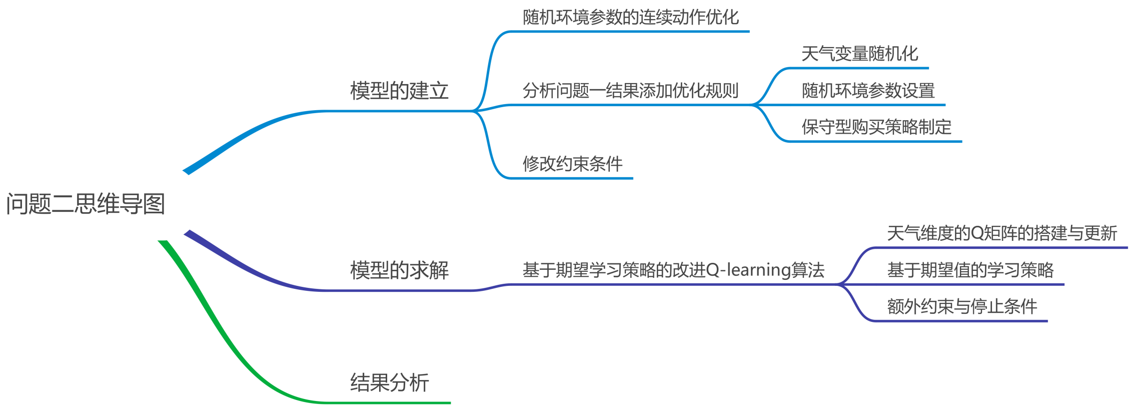


图 **6**　问题二思维流程图

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

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

其中 $s_{t,weather_t} \bigotimes a_t$ 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 $\{P(weather_t)(t = 1, 2, \cdots, n)\}$ 彼此之间相互独立，即可推得

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

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

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

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

$$\begin{cases} w(t_v) = \sum\limits_{t=t_v}^{n} \max\limits_{weather_t} \Delta w(weather_t) \cdot \lambda(s_t, a_t), \\ f(t_v) = \sum\limits_{t=t_v}^{n} \max\limits_{weather_t} \Delta f(weather_t) \cdot \lambda(s_t, a_t). \end{cases} \tag{14}$$

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

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

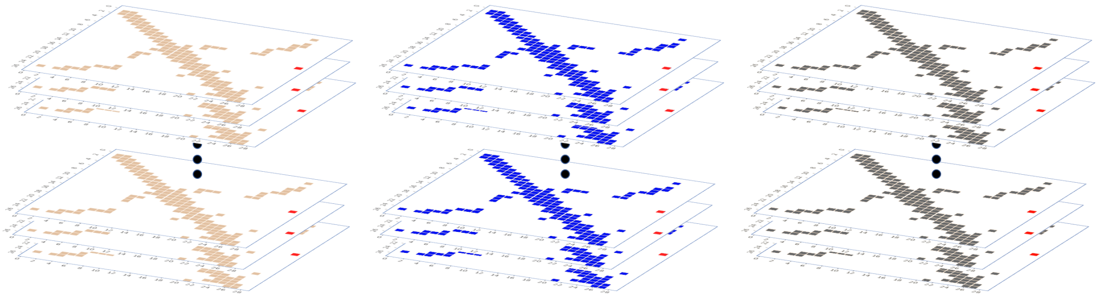
$$
s.t.\quad\begin{cases}
1 \leq n \leq T, \\
s_1 = s_{outset}, \\
s_{n+1} = s_{destination}, \\
|a_t| \leqslant weather_t, \\
\forall t = 0,1,2,...,T : w(t) \geqslant 0, \\
\forall t = 0,1,2,...,T : f(t) \geqslant 0, \\
\forall t = 0,1,2,...,T : w(t) + f(t) \leq b_0, \\
w(t+1) = w(t) - \Delta w(weather_t) \cdot \lambda(s_t, a_t), \\
f(t+1) = f(t) - \Delta f(weather_t) \cdot \lambda(s_t, a_t), \\
w(0) = \sum_{t=1}^{n} \max_{weather_t} \Delta w(weather_t) \cdot \lambda(s_t, a_t), \\
f(0) = \sum_{t=1}^{n} \max_{weather_t} \Delta f(weather_t) \cdot \lambda(s_t, a_t), \\
w(t_v) = \sum_{t=t_v}^{n} \max_{weather_t} \Delta w(weather_t) \cdot \lambda(s_t, a_t), \\
f(t_v) = \sum_{t=t_v}^{n} \max_{weather_t} \Delta f(weather_t) \cdot \lambda(s_t, a_t), \\
P(s_{t+1,weather_{t+1}} = s_{t,weather_t} \bigotimes a_t) = P(weather_{t+1}), \\
R(s_t, a_t, weather_t) = -consume(weather_t) \cdot \lambda(s_t, a_t) + income(s_t, a_t).
\end{cases}
\tag{16}
$$

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



(a) 沙暴概率 Q 矩阵 p1

(b) 高温概率 Q 矩阵 p2

(c) 晴天概率 Q 矩阵 p3

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

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

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

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

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

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

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。

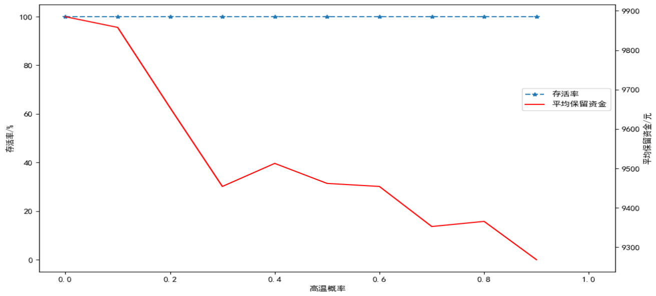
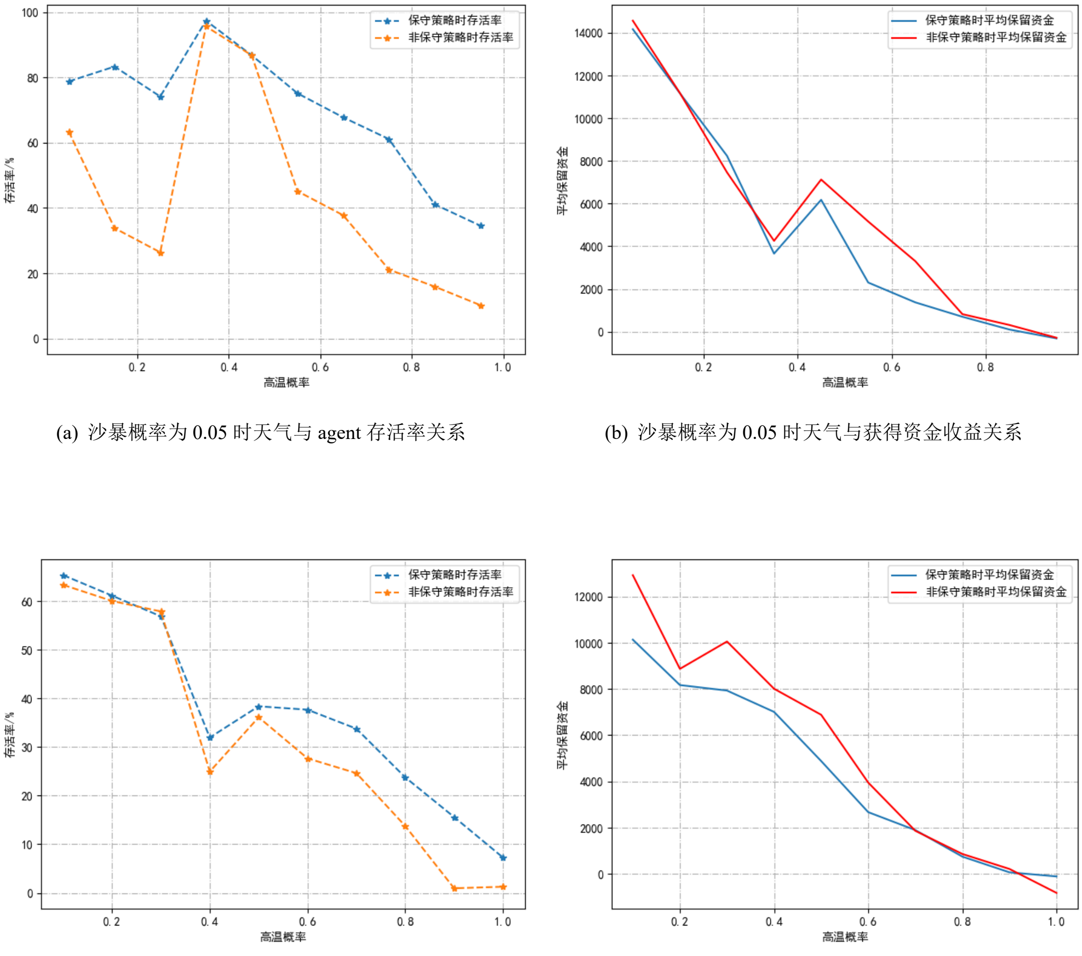


图 **8**　第三问学习策略结果仿真实验图

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 $u(n_1, n_2, ..., n_m)$，满足纯策略 $Nash$ 均衡

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

或表示为

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

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：

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

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

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

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

根据效用函数关系，有

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

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

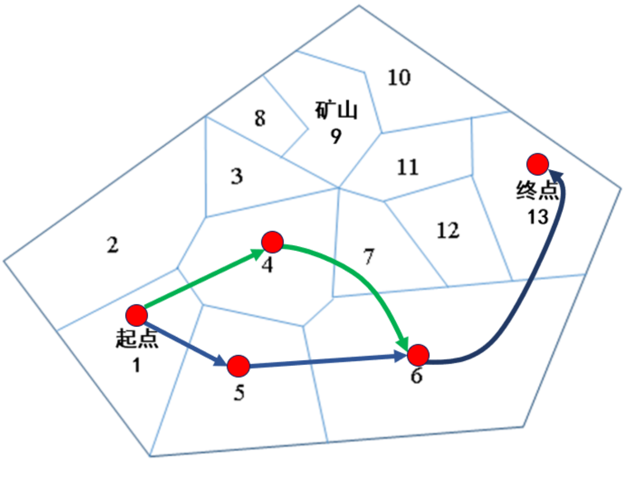


图 **10**　第五关双玩家策略路径图

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。

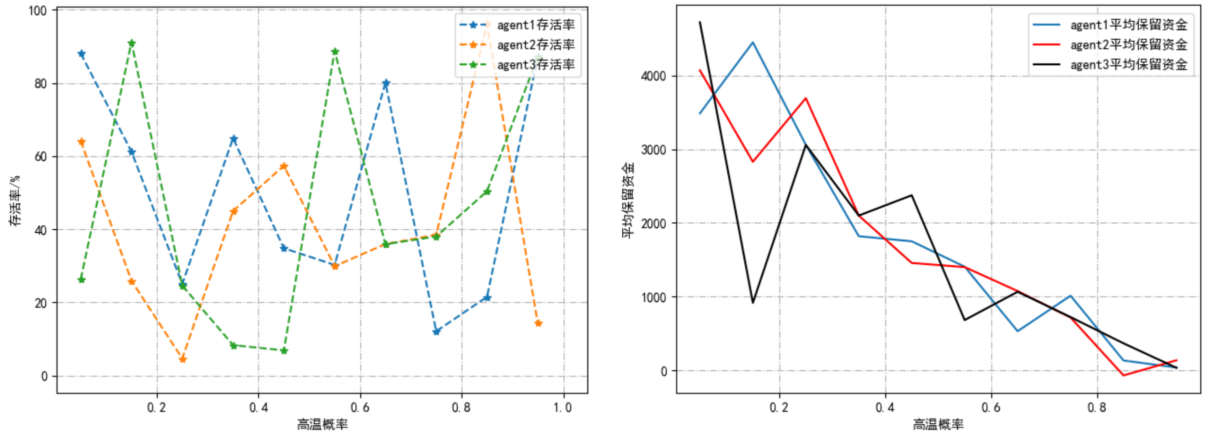
6.2.2 谦让策略

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

$$V_n(t) = Valve(w_n(t)) + Valve(f_n(t)) + M_n(t), \tag{22}$$

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 $n_{best} = \arg_n \max V_n(t)$、$n_{worst} = \arg_n \min V_n(t)$ 剩余的玩家为 n_{midle}。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 $n_{best} > n_{midle} > n_{worst}$，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



(a) 沙暴概率为 0.1 时天气与 agent 存活率关系

(b) 沙暴概率为 0.1 时天气与获得资金收益关系

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-learing 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-learing 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

```
import numpy as np
from numpy import isnan
import matplotlib.pyplot as plt
import seaborn as sns

def get_maze(checkpointnum):
    """
    :param checkpointnum: 关卡数目
    :return: 迷宫，天气
    """
    maze = []
    weather = []  # 1晴天，2高温，3沙暴
    assert checkpointnum in list(range(1,7))
    if checkpointnum==1:
        weather = [2,2,1,3,1,2,3,1,2,2,
                   3,2,1,2,2,2,3,3,2,2,
                   1,1,2,1,3,2,1,1,2,2]
        maze = np.full((27+1, 28+1), np.nan) # 28为挖，+1为0的地方不要
        maze[12,28] = 0   # 挖矿
        for i in range(1, 27+1):    # 停一天
            maze[i, i] = 0
        # 临边
        maze[1, 25],maze[1, 2]=0,0
        maze[2, 1],maze[2, 3]=0,0
        maze[3, 2],maze[3, 25],maze[3, 4]=0,0,0
        maze[4, 3],maze[4, 25],maze[4, 24],maze[4, 5]=0,0,0,0
        maze[5, 4],maze[5, 24],maze[5, 6]=0,0,0
        maze[6, 5],maze[6, 24],maze[6, 23],maze[6, 7]=0,0,0,0
        maze[7, 6],maze[7, 22],maze[7, 8]=0,0,0
        maze[8, 7],maze[8, 22],maze[8, 9]=0,0,0
        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
        maze[10, 9],maze[10, 15],maze[10, 13],maze[10, 11]=0,0,0,0
        maze[11, 10],maze[11, 13],maze[11, 12]=0,0,0
        maze[12, 11],maze[12, 13],maze[12, 14]=0,0,0
        maze[13, 10],maze[13, 15],maze[13, 14],maze[13, 12],maze[13, 11]=0,0,0,0,0
        maze[14, 15],maze[14, 16],maze[14, 12],maze[14, 13]=0,0,0,0
        maze[15, 10],maze[15, 9],maze[15, 16],maze[15, 14],maze[15, 13]=0,0,0,0,0
        maze[16, 14],maze[16, 15],maze[16, 9],maze[16, 17],maze[16, 18]=0,0,0,0,0
        maze[17, 9],maze[17, 21],maze[17, 18],maze[17, 16]=0,0,0,0
        maze[18, 17],maze[18, 20],maze[18, 19],maze[18, 16]=0,0,0,0
        maze[19, 18],maze[19, 20]=0,0
        maze[20, 21],maze[20, 18],maze[20, 19]=0,0,0
        maze[21, 27],maze[21, 23],maze[21, 22],maze[21, 9],maze[21, 17],maze[21, 20]=0,0,0,0,0,0
        maze[22, 7],maze[22, 23],maze[22, 21],maze[22, 9],maze[22, 8]=0,0,0,0,0
        maze[23, 24],maze[23, 26],maze[23, 21],maze[23, 22],maze[23, 6]=0,0,0,0,0
        maze[24, 5],maze[24, 6],maze[24, 23],maze[24, 26],maze[24, 25],maze[24, 4]=0,0,0,0,0,0
        maze[25, 1],maze[25, 24],maze[25, 3],maze[25, 4],maze[25, 26]=0,0,0,0,0
        maze[26, 25],maze[26, 24],maze[26, 23],maze[26, 27]=0,0,0,0
        maze[27, 26],maze[27, 21]=0,0
        pass
    elif checkpointnum==2:
        weather = [2, 2, 1, 3, 1, 2, 3, 1, 2, 2,
                   3, 2, 1, 2, 2, 2, 3, 3, 2, 2,
                   1, 1, 2, 1, 3, 2, 1, 1, 2, 2]
        maze = np.full((64 + 1, 65 + 1), np.nan)
        maze[30,65] = 0  # 挖矿
        maze[55,65] = 0  # 挖矿
        for i in range(1, 64+1):
            maze[i, i] = 0 # 停一天
            paishu = int((i-1)/8)+1 # 第几排
            tou = (paishu - 1) * 8 + 1 # 头
            wei = paishu * 8     # 尾
            if paishu==1:
                if i==tou:
                    maze[tou,tou+1],maze[tou,tou+8]=0,0
                elif i==wei:
                    maze[wei,wei-1],maze[wei,wei+7],maze[wei,wei+8]=0,0,0
                else:
                    maze[i,i-1],maze[i,i+1],maze[i,i+7],maze[i,i+8]=0,0,0,0
            if paishu==8:
                if i==tou:
                    maze[tou,tou-8],maze[tou,tou-7],maze[tou,tou+1]=0,0,0
                elif i==wei:
                    maze[wei,wei-1],maze[wei,wei-8]=0,0
                else:
                    maze[i,i-1],maze[i,i+1],maze[i,i-8],maze[i,i-7]=0,0,0,0
            if paishu in [2,4,6]:
                if i==tou:
                    maze[tou,tou-8],maze[tou,tou-7],maze[tou,tou+1],maze[tou,tou+8],maze
                elif i==wei:
                    maze[wei,wei-1],maze[wei,wei-8],maze[wei,wei+8]=0,0,0
                else:
                    maze[i,i-1],maze[i,i+1],maze[i,i-8],maze[i,i-7],maze[i,i+8],maze[i,
            if paishu in [3,5,7]:
                if i==tou:
                    maze[tou,tou+1],maze[tou,tou-8],maze[tou,tou+8]=0,0,0
                elif i==wei:
                    maze[wei,wei-1],maze[wei,wei-9],maze[wei,wei-8],maze[wei,wei+8],maze
                else:
                    maze[i,i-1],maze[i,i+1],maze[i,i-8],maze[i,i-9],maze[i,i+8],maze[i,
        pass
    elif checkpointnum==3:
        weather = []
        maze = np.full((13 + 1, 14 + 1), np.nan)
        maze[9, 14] = 0 # 挖矿
        for i in range(1, 13 + 1):
            maze[i, i] = 0 # 停一天
        maze[1, 2],maze[1, 5],maze[1, 4]=0,0,0
        maze[2, 1],maze[2, 4],maze[2, 3]=0,0,0
        maze[3, 2],maze[3, 4],maze[3, 9],maze[3, 8]=0,0,0,0
        maze[4, 1],maze[4, 2],maze[4, 3],maze[4, 7],maze[4, 6],maze[4, 5]=0,0,0,0,0,0
        maze[5, 1],maze[5, 4],maze[5, 6]=0,0,0
        maze[6, 5],maze[6, 4],maze[6, 7],maze[6, 12],maze[6, 13]=0,0,0,0,0
        maze[7, 4],maze[7, 11],maze[7, 12],maze[7, 6]=0,0,0,0
        maze[8, 3],maze[8, 9]=0,0
        maze[9, 3],maze[9, 8],maze[9, 11],maze[9, 10]=0,0,0,0
        maze[10, 9],maze[10, 11],maze[10, 13]=0,0,0
        maze[11, 9],maze[11, 10],maze[11, 13],maze[11, 12],maze[11, 7]=0,0,0,0,0
        maze[12, 7],maze[12, 11],maze[12, 13],maze[12, 6]=0,0,0,0
        maze[13, 6],maze[13, 12],maze[13, 11],maze[13, 10]=0,0,0,0
        pass
    elif checkpointnum==4:
        weather = []
        maze = np.full((25 + 1, 26 + 1), np.nan)
        maze[18, 26] = 0 # 挖矿
        for i in range(1, 25 + 1):
            maze[i, i] = 0 # 停一天
            if i==1:
                maze[i,i+1],maze[i,i+5]=0,0
            elif i==5:
                maze[i,i-1],maze[i,i+5]=0,0
            elif i==21:
                maze[i,i+1],maze[i,i-5]=0,0
            elif i==25:
                maze[i,i-1],maze[i,i-5]=0,0
            elif i in [2,3,4]:
                maze[i,i-1],maze[i,i+1],maze[i,i+5]=0,0,0
            elif i in [22,23,24]:
                maze[i,i-1],maze[i,i+1],maze[i,i-5]=0,0,0
            elif i in [6,11,16]:
                maze[i,i-5],maze[i,i+1],maze[i,i+5]=0,0,0
            elif i in [10,15,20]:
                maze[i,i-5],maze[i,i-1],maze[i,i+5]=0,0,0
            else:
                maze[i, i-1], maze[i, i + 1], maze[i, i-5], maze[i, i+5] = 0, 0, 0, 0
        pass
    elif checkpointnum==5:
        weather = [1,2,1,1,1,1,2,2,2,2,]
        maze = np.full((13 + 1, 14 + 1), np.nan)
        maze[9, 14] = 0 # 挖矿
        for i in range(1, 13 + 1):
            maze[i, i] = 0 # 停一天
        maze[1, 2],maze[1, 5],maze[1, 4]=0,0,0
        maze[2, 1],maze[2, 4],maze[2, 3]=0,0,0
        maze[3, 2],maze[3, 4],maze[3, 9],maze[3, 8]=0,0,0,0
        maze[4, 1],maze[4, 2],maze[4, 3],maze[4, 7],maze[4, 6],maze[4, 5]=0,0,0,0,0,0
        maze[5, 1],maze[5, 4],maze[5, 6]=0,0,0
        maze[6, 5],maze[6, 4],maze[6, 7],maze[6, 12],maze[6, 13]=0,0,0,0,0
        maze[7, 4],maze[7, 11],maze[7, 12],maze[7, 6]=0,0,0,0
        maze[8, 3],maze[8, 9]=0,0
        maze[9, 3],maze[9, 8],maze[9, 11],maze[9, 10]=0,0,0,0
        maze[10, 9],maze[10, 11],maze[10, 13]=0,0,0
        maze[11, 9],maze[11, 10],maze[11, 13],maze[11, 12],maze[11, 7]=0,0,0,0,0
        maze[12, 7],maze[12, 11],maze[12, 13],maze[12, 6]=0,0,0,0
        maze[13, 6],maze[13, 12],maze[13, 11],maze[13, 10]=0,0,0,0
        pass
    elif checkpointnum==6:
        weather = []
        maze = np.full((25 + 1, 26 + 1), np.nan)
        maze[18, 26] = 0 # 挖矿
        for i in range(1, 25 + 1):
            maze[i, i] = 0 # 停一天
            if i==1:
                maze[i,i+1],maze[i,i+5]=0,0
            elif i==5:
                maze[i,i-1],maze[i,i+5]=0,0
            elif i==21:
                maze[i,i+1],maze[i,i-5]=0,0
            elif i==25:
                maze[i,i-1],maze[i,i-5]=0,0
            elif i in [2,3,4]:
                maze[i,i-1],maze[i,i+1],maze[i,i+5]=0,0,0
            elif i in [22,23,24]:
                maze[i,i-1],maze[i,i+1],maze[i,i-5]=0,0,0
            elif i in [6,11,16]:
                maze[i,i-5],maze[i,i+1],maze[i,i+5]=0,0,0
            elif i in [10,15,20]:
                maze[i,i-5],maze[i,i-1],maze[i,i+5]=0,0,0
            else:
                maze[i, i-1], maze[i, i + 1], maze[i, i-5], maze[i, i+5] = 0, 0, 0, 0
        pass
    else:
        print("输入关卡数错误")
        return None
    return maze, np.array(weather)


def is_duicheng(maze, num):
    maze = maze[:num, :num]
    print(maze.shape)
    for i in range(num):
        for j in range(num):
            if maze[i,j]!=maze[j,i] and not isnan(maze[j,i]):
                print("不是对称",i,j)




if __name__ =="__main__":
    maze,weather = get_maze(1)
    print(maze.shape)
    print(maze)
    is_duicheng(maze,maze.shape[0])
    # https://blog.csdn.net/qq_42554007/article/details/82624418
    cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True,dark=0.6, light=0.6)
    sns.heatmap(maze, linewidths=0.05, cbar = True, cmap=cmap) #
    plt.show()
```

A.2 第一关代码

第一关 Q-learing 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]

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

```python
ISREAD_MAZES = False
maze, weathers = get_maze(guanqia)
wakuang = maze.shape[0] # 终点位置
PATH = 'demo1'
#[25, 24, 23, 23, 22, 9, 9, 15, 13, 12, 28, 28, 28, 28, 28, 14, 14, 14, 14, 14, 15, 9, 21, 27]

# starts_path= [1]
# starts_A= []
starts_path = [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] # 开始路径
starts_A = [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] # 开始状态

# starts_path = [1,25, 24, 23, 23, 22, 9, 9, 15, 13, 12,12,12,12,12,12,12,12,12,13,15, 9, 21,] # 开始路径
# starts_A = [25, 24, 23, 23, 22, 9, 9, 15, 13, 12,12,28,28,28,28,28,28,28,13,15, 9, 21,] # 开始状态
SSSSS = starts_path[len(starts_path) - 1]# 起点


def get_pathmap(maze):
    path_map = []
    # 每个状态拥有的动作
    for i in range(maze.shape[0]): # 初始化所有可行域
        temp = []
        for j in range(1, maze.shape[1]):
            if not np.isnan(maze[i,j]):
                temp.append(j)
        path_map.append(temp)
    return path_map


def get_w_i(weather):
    if weather==1:
        w_i = 3*5+2*7
        p_i = 5*5+10*7
    elif weather==2:
        w_i = 3*8+2*6
        p_i = 5*8+10*6
    else:
        w_i = 3*10+2*10
        p_i = 5*10+10*10
    return w_i, p_i


def probability_fun(sss, max_a, probability=0.8, k=None):
    '''
    :param probability: 0为贪婪，1为随机策略，其他为sigema策略
    '''
    if probability==1:
        return random.choice(sss)
    elif probability==0:
        list_a = max_a.tolist()
        q_max = float('-inf')
        for i in sss:
            if list_a[i] > q_max and not np.isnan(list_a[i]):
                q_max=list_a[i]
        # if probability==0:
        #    print(q_max, list_a.index(q_max))
        return list_a.index(q_max)
    else:
        if k<yuzhi:
            probability=-(1-0.5)*k/yuzhi+1
        else:
            probability=0
        if np.random.random()<probability:
            return random.choice(sss)
        else:
            list_a = max_a.tolist()
            q_max = float('-inf')
            for i in sss:
                if list_a[i] > q_max and not np.isnan(list_a[i]):
                    q_max = list_a[i]
            # if probability==0:
            #    print(q_max, list_a.index(q_max))
            return list_a.index(q_max)


def get_W_status(starts_path, starts_A):
    s = 1 # 起点
    w = 0 # 背包负重
    w_chunzhuang = 0 # 村庄买的东西
    plan = [W_UP, 0] # 计划购买栈
    Rs = []
    flag = False  # 是否饿死
    for t in range(len(starts_path)-1):# 第0天就开始动，第30天没有动
        where = starts_path[t]
        s_p = starts_A[t] if starts_A[t] != wakuang else where#下一步要干嘛

        w_i, r_i = get_w_i(weathers[t]) # 第t天的基础消耗
        R = -r_i # 基础消耗
        w_i_i = w_i # 基础消耗
        if where!=s_p: # 走，2倍
            R = - 2 * r_i
            w_i_i = 2 * w_i
        if s_p == wakuang: # 挖矿钱
            R = -3 * r_i + shouyi
            w_i_i = 3 * w_i
        w = w + w_i_i
        w_chunzhuang = w_chunzhuang + w_i_i

        if plan[0] >= w_i_i: # 不需要村庄
            plan[0] = plan[0] - w_i_i
        else: # 需要用村庄
            plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
            plan[0] = 0
            if where!=s_p: # 走，2倍
                R = - 4 * r_i
            if s_p == wakuang: # 挖矿钱
                R = -6 * r_i + shouyi # 挖矿钱
            if plan[1] < 0:
                flag = True # 死了

        print("第{}天执行{}->{}".format(t,where,s_p), plan, w_chunzhuang,w)

        # 村庄买东西
        if where in cunzhuang: # 村庄
            plan[1] = plan[1] + w_chunzhuang
            w_chunzhuang = 0

        Rs.append(R)
    return plan,Rs,flag


def get_path(mazes, path_map):
    path = starts_path.copy()
    A = starts_A.copy()
    s = SSSSS
    for t in range(start_t, len(weathers)-1): # (10, 30)
        map = mazes[t,:,:]
        if s==wakuang - 1:
            break
        a = probability_fun(path_map[s], map[s,:], 0)
        s_p = a if a!=wakuang else s

        path.append(s_p)
        A.append(a)
```

```python
        s = s_p
    return path, A


if __name__ == '__main__':
    mazes = []    # 时间图Q
    for i in range(len(weathers)):
        mazes.append(maze)
    mazes = np.array(mazes).copy()
    path_map = get_pathmap(maze)
    print(mazes.shape) # mazes (时间，状态，动作)
    print(path_map)
    print(path_map[21])

    # R放入初值中
    # for t in range(len(weathers)):
    #    for j in range(wakuang): # 初始化所有可行域
    #        for a in path_map[j]:
    #            w_i, r_i = get_w_i(weathers[t])
    #            R = -r_i
    #            if a != j: # 走，2倍
    #                R = - 2 * r_i
    #            if a == maze.shape[1] - 1:
    #                R = -3 * r_i + shouyi # 挖矿钱
    #            mazes[t,j,a] = R

    #  mazes付初值
    if ISREAD_MAZES:
        mazes = np.array(np.load("./data/{}_{}.npz".format(PATH,guanqia))['data'])

    # 初值的可视化
    cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True, dark=0.7, light=0.3)
    sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True, cmap=cmap) #
```

```
plt.show()

print('######## Q-learing ########')
plansss, Rs, flag = get_W_status(starts_path, starts_A)
print(flag, plansss)
start_t = len(starts_path)-1
print("起点", SSSSS, "第几天", start_t, len(weathers)-1,"矿位置", wakuang)
for k in range(learn_num):# 迭代次数
    s = SSSSS # 起点

    plan = plansss.copy() # 计划购买栈
    path = starts_path.copy()
    w = 1200-plansss[0] # 背包负重
    w_chunzhuang = 1200-plansss[0] # 村庄买的东西
    A = starts_A.copy()
    flag = False
    # print("*"*50)
    # is_kuang = False
    # print(start_t, len(weathers))
    for t in range(start_t, len(weathers)-1):#(10, 30)
        a = probability_fun(path_map[s], mazes[t, s, :], k=k) # 动作
        # if s not in kuang and a in kuang:
        #     is_kuang=True
        # if is_kuang:
        #     a=s
        #     is_kuang = False

        s_p = a if a != wakuang else s # 下一步要干嘛

        w_i, r_i = get_w_i(weathers[t])
        R = -r_i # 走一步消耗
        w_i_i = w_i
        if a != s: # 走, 2倍
            R = - 2 * r_i
```

```
                w_i_i = 2 * w_i
            if a == wakuang:
                R = -3 * r_i + shouyi # 挖矿钱
                w_i_i = 3 * w_i
            if weathers[t] == 3: # 沙包必须停留
                s_p = s
                # if a!=wakuang and a!=s:
                #    a=s

            w = w + w_i_i
            w_chunzhuang = w_chunzhuang + w_i_i

            if plan[0] >= w_i_i: # 不需要村庄
                plan[0] = plan[0] - w_i_i
            else: # 需要用村庄
                plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
                plan[0] = 0
                if s != s_p: # 走，2倍
                    R = - 4 * r_i
                if a == wakuang: # 挖矿钱
                    R = -6 * r_i + shouyi # 挖矿钱
                if plan[1] < 0:
                    flag = True # 死了
            if plan[1] < 0:
                flag = True # 死了

            # print("第{}天执行{}, {}->{}".format(t,a,s,s_p), plan)

            # 村庄买东西
            if s in cunzhuang: # 村庄
                plan[1] = plan[1] + w_chunzhuang
                w_chunzhuang = 0

            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
```

```python

        if s_p == wakuang - 1: # 终点
            mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
            s = s_p # 更新位置
            # print(path)
            break      # 29
        elif flag: # 中止条件
            R = R - 1000000
            mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
            s = s_p # 更新位置
            break
        elif t == len(weathers) - 1 -1 and s_p != wakuang-1:
            R = R - 1000000
            mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
            s = s_p # 更新位置
            break
        else:
            mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
            s = s_p # 更新位置

        A.append(a)
        path.append(s)
    # break

np.savez_compressed("./data/{}_{}.npz".format(PATH,guanqia), data=mazes)

path, A = get_path(mazes, path_map)
print(path)
print(A)
#
#
```

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

A.3 第二关代码

第二关 Q-learing 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]

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

# starts_path=
    [1,9,10,19,19,27,36,36,44,53,54,62,55,55,55,55,55,55,55,55,62,55,55]
# starts_A=
    [9,10,19,19,27,36,36,44,53,54,62,55,65,65,65,65,65,65,65,62,55,65]
# starts_path= [1, 2, 10, 11, 11, 12, 13, 13, 22, 30]
# starts_A= [ 2, 10, 11, 11, 12, 13, 13, 22, 30]
#
starts_path= [1, 9, 10, 19, 19, 27, 36, 36, 44, 53, 54, 54, 62,
              55, 65, 65, 65, 65, 65, 65, 65,65, 62, 55, 65, 65, 65, 65,
                  65, 56, 64]
starts_A = [9, 10, 19, 19, 27, 36, 36, 44, 53, 54, 54, 62, 55,
            65, 65, 65, 65, 65, 65, 65,65, 62, 55, 65, 65, 65, 65, 65,
                56, 64]

SSSSS = starts_path[len(starts_path) - 1]# 起点


def get_pathmap(maze):
    path_map = []
    # 每个状态拥有的动作
    for i in range(maze.shape[0]): # 初始化所有可行域
        temp = []
        for j in range(1, maze.shape[1]):
            if not np.isnan(maze[i,j]):
                temp.append(j)
        path_map.append(temp)
    return path_map


def get_w_i(weather):
    if weather==1:
        w_i = 3*5+2*7
        p_i = 5*5+10*7
    elif weather==2:
        w_i = 3*8+2*6
        p_i = 5*8+10*6
    else:
        w_i = 3*10+2*10
        p_i = 5*10+10*10
    return w_i, p_i


def probability_fun(sss, max_a, probability=0.8, k=None):
    '''
    :param probability: 0为贪婪，1为随机策略，其他为sigema策略
    '''
    if probability==1:
        return random.choice(sss)
    elif probability==0:
        list_a = max_a.tolist()
        q_max = float('-inf')
        for i in sss:
            if list_a[i] > q_max and not np.isnan(list_a[i]):
                q_max=list_a[i]
        # if probability==0:
        #    print(q_max, list_a.index(q_max))
        return list_a.index(q_max)
    else:
        if k<yuzhi:
            probability=-(1-0.5)*k/yuzhi+1
        else:
            probability=0
        if np.random.random()<probability:
            return random.choice(sss)
        else:
            list_a = max_a.tolist()
            q_max = float('-inf')
            for i in sss:
                if list_a[i] > q_max and not np.isnan(list_a[i]):
                    q_max = list_a[i]
            # if probability==0:
            #    print(q_max, list_a.index(q_max))
            return list_a.index(q_max)


def get_W_status(starts_path, starts_A):
    s = 1 # 起点
    w = 0 # 背包负重
    w_chunzhuang = 0 # 村庄买的东西
    plan = [W_UP, 0] # 计划购买栈
    Rs = []
    flag = False  # 是否饿死
    for t in range(len(starts_path)-1):# 第0天就开始动，第30天没有动
        where = starts_path[t]
        s_p = starts_A[t] if starts_A[t] != wakuang else
            where#下一步要干嘛

        w_i, r_i = get_w_i(weathers[t]) # 第t天的基础消耗
        R = -r_i # 基础消耗
        w_i_i = w_i # 基础消耗
        if where!=s_p: # 走，2倍
            R = - 2 * r_i
            w_i_i = 2 * w_i
        if s_p == wakuang: # 挖矿钱
            R = -3 * r_i + shouyi
            w_i_i = 3 * w_i
        w = w + w_i_i
        w_chunzhuang = w_chunzhuang + w_i_i

        if plan[0] >= w_i_i: # 不需要村庄
            plan[0] = plan[0] - w_i_i
        else: # 需要用村庄
            plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
            plan[0] = 0
            if where!=s_p: # 走，2倍
```

```
                R = - 4 * r_i
            if s_p == wakuang: # 挖矿钱
                R = -6 * r_i + shouyi # 挖矿钱
            if plan[1] < 0:
                flag = True # 死了

        print("第{}天执行{}->{}".format(t,where,s_p), plan, w_chunzhuang,w)

        # 村庄买东西
        if where in cunzhuang: # 村庄
            plan[1] = plan[1] + w_chunzhuang
            w_chunzhuang = 0

        Rs.append(R)
    return plan,Rs,flag


def get_path(mazes, path_map):
    path = starts_path.copy()
    A = starts_A.copy()
    s = SSSSS
    for t in range(start_t, len(weathers)-1): # (10, 30)
        map = mazes[t,:,:]
        if s==wakuang - 1:
            break
        a = probability_fun(path_map[s], map[s,:], 0)
        s_p = a if a!=wakuang else s

        path.append(s_p)
        A.append(a)
        s = s_p
    return path, A


```

```
if __name__ == '__main__':
    mazes = []    # 时间图Q
    for i in range(len(weathers)):
        mazes.append(maze)
    mazes = np.array(mazes).copy()
    path_map = get_pathmap(maze)
    print(mazes.shape) # mazes (时间, 状态, 动作)
    print(path_map)
    print(path_map[21])

    # R放入初值中
    # for t in range(len(weathers)):
    #    for j in range(wakuang): # 初始化所有可行域
    #        for a in path_map[j]:
    #            w_i, r_i = get_w_i(weathers[t])
    #            R = -r_i
    #            if a != j: # 走, 2倍
    #                R = - 2 * r_i
    #            if a == maze.shape[1] - 1:
    #                R = -3 * r_i + shouyi # 挖矿钱
    #            mazes[t,j,a] = R

    #  mazes付初值
    if ISREAD_MAZES:
        mazes = np.array(np.load("./data/{}_{}.npz".format(PATH,guanqia))['data'])

    # 初值的可视化
    cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True, dark=0.7, light=0.3)
    sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True, cmap=cmap) #
    plt.show()

    print('######## Q-learing ########')
    plansss, Rs, flag = get_W_status(starts_path, starts_A)
```

```python
    print(flag, plansss)
    start_t = len(starts_path)-1
    print("起点", SSSSS, "第几天", start_t, len(weathers)-1,"矿位置",
        wakuang)
    for k in range(learn_num):# 迭代次数
        s = SSSSS # 起点

        plan = plansss.copy() # 计划购买栈
        path = starts_path.copy()
        w = 1200-plansss[0] # 背包负重
        w_chunzhuang = 1200-plansss[0] # 村庄买的东西
        A = starts_A.copy()
        flag = False
        # print("*"*50)
        # is_kuang = False
        # print(start_t, len(weathers))
        for t in range(start_t, len(weathers)-1):#(10, 30)
            a = probability_fun(path_map[s], mazes[t, s, :], k=k) # 动作
            # if s not in kuang and a in kuang:
            #    is_kuang=True
            # if is_kuang:
            #    a=s
            #    is_kuang = False

            s_p = a if a != wakuang else s # 下一步要干嘛

            w_i, r_i = get_w_i(weathers[t])
            R = -r_i # 走一步消耗
            w_i_i = w_i
            if a != s: # 走，2倍
                R = - 2 * r_i
                w_i_i = 2 * w_i
            if a == wakuang:
                R = -3 * r_i + shouyi # 挖矿钱
                w_i_i = 3 * w_i
            if weathers[t] == 3: # 沙包必须停留
                s_p = s
                # if a!=wakuang and a!=s:
                #     a=s

            w = w + w_i_i
            w_chunzhuang = w_chunzhuang + w_i_i

            if plan[0] >= w_i_i: # 不需要村庄
                plan[0] = plan[0] - w_i_i
            else: # 需要用村庄
                plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
                plan[0] = 0
                if s != s_p: # 走，2倍
                    R = - 4 * r_i
                if a == wakuang: # 挖矿钱
                    R = -6 * r_i + shouyi # 挖矿钱
                if plan[1] < 0:
                    flag = True # 死了
            if plan[1] < 0:
                flag = True # 死了

            # print("第{}天执行{}, {}->{}".format(t,a,s,s_p), plan)

            # 村庄买东西
            if s in cunzhuang: # 村庄
                plan[1] = plan[1] + w_chunzhuang
                w_chunzhuang = 0

            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

            if s_p == wakuang - 1: # 终点
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
                s = s_p # 更新位置
                # print(path)
                break      # 29
            elif flag: # 中止条件
                R = R - 1000000
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
                s = s_p # 更新位置
                break
            elif t == len(weathers) - 1 -1 and s_p != wakuang-1:
                R = R - 1000000
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
                s = s_p # 更新位置
                break
            else:
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
                s = s_p # 更新位置

            A.append(a)
            path.append(s)
        # break
    np.savez_compressed("./data/{}_{}.npz".format(PATH,guanqia), data=mazes)

    path, A = get_path(mazes, path_map)
    print(path)
    print(A)
    #
    #
    # for t in range(start_t, len(weathers)):
    #    map = mazes[t,:,:]
    #    for i in range(map.shape[0]):
    #        for j in range(map.shape[1]):
    #            if map[i,j]<-100000 and not np.isnan(map[i,j]):
    #               map[i, j] = -1000000
    #    cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True, dark=0.7, light=0.3)
    #    sns.heatmap(map, linewidths=0.05, cbar=True, cmap=cmap) #
    #    plt.show()
```

A.4 第三关代码

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

```
import numpy as np
from maze import get_maze
import random
import matplotlib.pyplot as plt
import seaborn as sns

jilu_jiage = []

# ISREAD_MAZES = True

guanqia = 3 #关卡
learn_num = 100000 #迭代次数
W_UP = 1200  #负重上线
aleph, gamma = 0.8, 0.8 #学习率
shouyi = 200 # 挖矿收益
ddd = 10   # 天数
cunzhuang = []
kuang = [9]      # 矿山
maze, _ = get_maze(guanqia)
wakuang = maze.shape[0] # 终点位置
# PATH = 'demo4'
starts_path= [1]
starts_A= []
start_t = len(starts_path) - 1
SSSSS = starts_path[len(starts_path) - 1]# 起点
shabao = 0
```

```python
def get_weather(gaowengailv):
    weatherssss = []
    for i in range(ddd):
        x = random.random()
        if x<shabao:
            weatherssss.append(3)
        elif x>=shabao and x<gaowengailv:
            weatherssss.append(2)
        else:
            weatherssss.append(1)
    return weatherssss


def get_w_i(weather):
    w_i, p_i = 0, 0
    if weather==1:
        w_i = 3*3+2*4
        p_i = 3*5+10*4
    elif weather==2:
        w_i = 9*3+2*9
        p_i = 9*5+10*9
    else:
        w_i = 3*10+2*10
        p_i = 5*10+10*10
    return w_i, p_i


def get_pathmap(maze):
    path_map = []
    # 每个状态拥有的动作
    for i in range(maze.shape[0]): # 初始化所有可行域
        temp = []
        for j in range(1, maze.shape[1]):
            if not np.isnan(maze[i,j]):
                temp.append(j)
        path_map.append(temp)
    return path_map


def probability_fun(sss, max_a, probability=0.8, k=None):
    '''
    :param probability: 0为贪婪，1为随机策略，其他为sigema策略
    '''
    if probability==1:
        return random.choice(sss)
    elif probability==0:
        list_a = max_a.tolist()
        q_max = float('-inf')
        for i in sss:
            if list_a[i] > q_max and not np.isnan(list_a[i]):
                q_max=list_a[i]
        # if probability==0:
        #    print(q_max, list_a.index(q_max))
        return list_a.index(q_max)
    else:
        if k<yuzhi:
            probability=-(1-0.5)*k/yuzhi+1
        else:
            probability=0
        if np.random.random()<probability:
            return random.choice(sss)
        else:
            list_a = max_a.tolist()
            q_max = float('-inf')
            for i in sss:
                if list_a[i] > q_max and not np.isnan(list_a[i]):
                    q_max = list_a[i]
            # if probability==0:
```

```
            #    print(q_max, list_a.index(q_max))
            return list_a.index(q_max)


def get_W_status(starts_path, starts_A, weathers):
    s = 1 # 起点
    w = 0 # 背包负重
    w_chunzhuang = 0 # 村庄买的东西
    plan = [W_UP, 0] # 计划购买栈
    Rs = []
    flag = False  # 是否饿死
    for t in range(len(starts_path)-1):# 第0天就开始动，第30天没有动
        where = starts_path[t]
        s_p = starts_A[t] if starts_A[t] != wakuang else where#下一步要干嘛

        w_i, r_i = get_w_i(weathers[t]) # 第t天的基础消耗
        R = -r_i # 基础消耗
        w_i_i = w_i # 基础消耗
        if where!=s_p: # 走，2倍
            R = - 2 * r_i
            w_i_i = 2 * w_i
        if s_p == wakuang: # 挖矿钱
            R = -3 * r_i + shouyi
            w_i_i = 3 * w_i
        w = w + w_i_i
        w_chunzhuang = w_chunzhuang + w_i_i

        if plan[0] >= w_i_i: # 不需要村庄
            plan[0] = plan[0] - w_i_i
        else: # 需要用村庄
            plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
            plan[0] = 0
            if where!=s_p: # 走，2倍
                R = - 4 * r_i
            if s_p == wakuang: # 挖矿钱
                R = -6 * r_i + shouyi # 挖矿钱
            if plan[1] < 0:
                flag = True # 死了

        print("第{}天执行{}->{}".format(t,where,s_p), plan, w_chunzhuang,w)

        # 村庄买东西
        if where in cunzhuang: # 村庄
            plan[1] = plan[1] + w_chunzhuang
            w_chunzhuang = 0

        Rs.append(R)
    return plan,Rs,flag


def get_path(mazes, path_map, weathers):
    path = starts_path.copy()
    A = starts_A.copy()
    s = SSSSS
    for t in range(start_t, len(weathers)-1): # (10, 30)
        map = mazes[t,:,:]
        if s==wakuang - 1:
            break
        a = probability_fun(path_map[s], map[s,:], 0)
        s_p = a if a!=wakuang else s

        path.append(s_p)
        A.append(a)
        s = s_p
    return path, A


def get_x_y(weather):
```

```python
    if weather==1:
        x = 3
        y = 4
    elif weather==2:
        x = 9
        y = 9
    else:
        x = 10
        y = 10
    return x, y


def get_R_x_ys(path,weather,sss):
    zhuan = 0
    for i in path:
        if i==sss:
            zhuan = zhuan+1000
    xs,ys=[],[]
    s=1
    for t in range(1,len(path)):
        a,b = get_x_y(weather[t-1])
        s_p = path[t]
        # s_p=get_index(s,a_pp)
        A=a
        B=b
        if s_p != s and s_p!=sss: # 走, 2倍
            A=2*a
            B=2*b
        if s_p == sss:
            A = 3 * a
            B = 3 * b
        xs.append(A)
        ys.append(B)
        # print(s,"->>",s_p," ", A,B)
        s = s_p

    x=sum(xs)
    y=sum(ys)
    R = int((1200 - 2 * y)/3) * 5 + (x + y - int((1200 - 2 * y)/3)) * 10 - zhuan
    return -R,xs,ys,zhuan


def get_awser(maze,path, weather):
    s = 1 # 起点
    w = 0 # 背包负重
    w_chunzhuang = 0 # 村庄买的东西
    plan = [W_UP, 0] # 计划购买栈
    Rs = []
    flag = False
    for t in range(1,len(path)):
        w_i, r_i = get_w_i(weather[t-1])
        R = -r_i # 走一步消耗
        w_i_i = w_i
        if path[t] != s: # 走，2倍
            R = - 2 * r_i
            w_i_i = 2 * w_i
        if path[t] == maze.shape[1]-1:
            R = -3 * r_i + shouyi # 挖矿钱
            w_i_i = 3 * w_i
        w = w + w_i_i
        w_chunzhuang = w_chunzhuang + w_i_i

        if plan[0] >= w_i_i: # 不需要村庄
            plan[0] = plan[0] - w_i_i
            pass
        else: # 需要用村庄
            plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
            plan[0] = 0
            if path[t] != s: # 走，2倍
```

```
                R = - 4 * r_i
            if path[t] == maze.shape[1]-1:
                R = -6 * r_i + shouyi # 挖矿钱
        if plan[1] < 0:
            flag = True
        # print(t,s,plan, w_chunzhuang,w)

        # 村庄买东西
        if s in cunzhuang: # 村庄
            plan[1] = plan[1] + w_chunzhuang
            w_chunzhuang = 0

        s = path[t]
        Rs.append(R)
    return plan,Rs,flag


def main(gaowengailv, ISREAD_MAZES, PATH):
    mazes = []    # 时间图Q
    weathers = get_weather(gaowengailv)
    for i in range(len(weathers)):
        mazes.append(maze)
    mazes = np.array(mazes).copy()
    path_map = get_pathmap(maze)
    # print(mazes.shape) # mazes (时间, 状态, 动作)
    # print(path_map)
    # print(path_map[21])

    #  mazes付初值
    if ISREAD_MAZES:
        mazes =
            np.array(np.load("./data/{}_{}.npz".format(PATH,guanqia))['data'])
    # 初值的可视化
    #    cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True,
        dark=0.7, light=0.3)
```

```
#    sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True, cmap=cmap) #
#    plt.show()
mazes_demo = mazes.copy()

print('######## Q-learing ########')
plansss, Rs, flag = get_W_status(starts_path, starts_A, weathers)
# print(flag, plansss)
print("起点", SSSSS, "第几天", start_t, len(weathers)-1,"矿位置", wakuang)

huos = 0
jiage = []
for k in range(learn_num):# 迭代次数
    s = SSSSS # 起点

    plan = plansss.copy() # 计划购买栈
    path = starts_path.copy()
    w = 1200-plansss[0] # 背包负重
    w_chunzhuang = 1200-plansss[0] # 村庄买的东西
    A = starts_A.copy()
    flag = False
    huo = 0
    # print("*"*50)
    # is_kuang = False
    # print(start_t, len(weathers))
    if ISREAD_MAZES:
        mazes = mazes_demo.copy()
        weathers = get_weather(gaowengailv).copy()

    for t in range(start_t, len(weathers)-1):#(10, 30)

        a = probability_fun(path_map[s], mazes[t, s, :], k=k) # 动作
        # if s not in kuang and a in kuang:
        #    is_kuang=True
        # if is_kuang:
        #    a=s
        #    is_kuang = False

        s_p = a if a != wakuang else s # 下一步要干嘛

        w_i, r_i = get_w_i(weathers[t])
        R = -r_i # 走一步消耗
        w_i_i = w_i
        if a != s: # 走，2倍
            R = - 2 * r_i
            w_i_i = 2 * w_i
        if a == wakuang:
            R = -3 * r_i + shouyi # 挖矿钱
            w_i_i = 3 * w_i
        if weathers[t] == 3: # 沙包必须停留
            s_p = s
            # if a!=wakuang and a!=s:
            #    a=s

        w = w + w_i_i
        w_chunzhuang = w_chunzhuang + w_i_i

        if plan[0] >= w_i_i: # 不需要村庄
            plan[0] = plan[0] - w_i_i
        else: # 需要用村庄
            plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
            plan[0] = 0
            if s != s_p: # 走，2倍
                R = - 4 * r_i
            if a == wakuang: # 挖矿钱
                R = -6 * r_i + shouyi # 挖矿钱
            if plan[1] < 0:
                flag = True # 死了
        if plan[1] < 0:
            flag = True # 死了

        # print("第{}天执行{}, {}->{}".format(t,a,s,s_p), plan)

        # 村庄买东西
        if s in cunzhuang: # 村庄
            plan[1] = plan[1] + w_chunzhuang
            w_chunzhuang = 0

        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

        if s_p == wakuang - 1: # 终点
            huo = 1
            mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
            s = s_p # 更新位置
            # print(path)
            break     # 29
        elif flag: # 中止条件
            R = R - 1000000
            mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
            s = s_p # 更新位置
            break
        elif t == len(weathers) - 1 -1 and s_p != wakuang-1:
            R = R - 1000000
            mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
            s = s_p # 更新位置
            break
        else:
            mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
            s = s_p # 更新位置
        A.append(a)
```

```
            path.append(s)
        # break


        A.append(25)
        pathsss = [1]
        for i in A:
            pathsss.append(i)
        if huo:
            huos = huos + 1 # 活着
            R, xs, ys, zhuan = get_R_x_ys(pathsss, weathers, 26)
            # print(R)
            # print(pathsss)
            jiage.append(R)

    # if ISREAD_MAZES:
    #    assert len(jiage)==huos
    print("平均价格", sum(jiage)/huos, '活着概率', round(huos*100/learn_num,2),"%")
    np.savez_compressed("./data/{}_{}.npz".format(PATH,guanqia), data=mazes)

    path, A = get_path(mazes, path_map, weathers)
    # print(path)
    # print(A)
    paths = [1]
    for i in A:
        paths.append(i)

    # print(len(paths))
    plan,Rs, flag = get_awser(maze, paths, weathers)
    print(paths)
    # print(plan)
    # print(Rs)
    print("****************************")
    print(flag)
    R, xs, ys, zhuan = get_R_x_ys(paths, weathers, 26)
    print(R)


plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签


if __name__ == '__main__':
    gaowengailv = 0.1
    ISREAD_MAZES = False
    yuzhi = int(learn_num * 0.9) if not ISREAD_MAZES else 0 # 策略域

    gaowengailv2 = gaowengailv + shabao
    main(gaowengailv2, ISREAD_MAZES,
        'demo3_gailv_1_01_{}'.format(int(gaowengailv * 100)))

    ISREAD_MAZES = True
    yuzhi = int(learn_num * 0.9) if not ISREAD_MAZES else 0 # 策略域
    main(gaowengailv2, ISREAD_MAZES,
        '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-learing 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。

```
import numpy as np
from maze import get_maze
import random
import matplotlib.pyplot as plt
import seaborn as sns

jilu_jiage = []

```

```python
# ISREAD_MAZES = True

guanqia = 4 #关卡
learn_num = 100000 #迭代次数
W_UP = 1200  #负重上线
aleph, gamma = 0.8, 0.8 #学习率
shouyi = 1000 # 挖矿收益
ddd = 30  # 天数
cunzhuang = [14]
kuang = [18]      # 矿山
maze, _ = get_maze(guanqia)
wakuang = maze.shape[0] # 终点位置
# PATH = 'demo4'
starts_path= [1]
starts_A= []
start_t = len(starts_path) - 1
SSSSS = starts_path[len(starts_path) - 1]# 起点
# gaowengailv = 0.3 + 0.05 # 高温概率(其中0.05是沙暴概率，固定)
shabao = 0.05


def get_weather(gaowengailv):
    weatherssss = []
    for i in range(ddd):
        x = random.random()
        if x<shabao:
            weatherssss.append(3)
        elif x>=shabao and x<gaowengailv:
            weatherssss.append(2)
        else:
            weatherssss.append(1)
    return weatherssss


def get_w_i(weather):
    w_i, p_i = 0, 0
    if weather==1:
        w_i = 3*3+2*4
        p_i = 3*5+10*4
    elif weather==2:
        w_i = 9*3+2*9
        p_i = 9*5+10*9
    else:
        w_i = 3*10+2*10
        p_i = 5*10+10*10
    return w_i, p_i


def get_pathmap(maze):
    path_map = []
    # 每个状态拥有的动作
    for i in range(maze.shape[0]): # 初始化所有可行域
        temp = []
        for j in range(1, maze.shape[1]):
            if not np.isnan(maze[i,j]):
                temp.append(j)
        path_map.append(temp)
    return path_map


def probability_fun(sss, max_a, probability=0.8, k=None):
    '''
    :param probability: 0为贪婪，1为随机策略，其他为sigema策略
    '''
    if probability==1:
        return random.choice(sss)
    elif probability==0:
        list_a = max_a.tolist()
        q_max = float('-inf')
        for i in sss:
            if list_a[i] > q_max and not np.isnan(list_a[i]):
                q_max=list_a[i]
        # if probability==0:
        #     print(q_max, list_a.index(q_max))
        return list_a.index(q_max)
    else:
        if k<yuzhi:
            probability=-(1-0.5)*k/yuzhi+1
        else:
            probability=0
        if np.random.random()<probability:
            return random.choice(sss)
        else:
            list_a = max_a.tolist()
            q_max = float('-inf')
            for i in sss:
                if list_a[i] > q_max and not np.isnan(list_a[i]):
                    q_max = list_a[i]
            # if probability==0:
            #     print(q_max, list_a.index(q_max))
            return list_a.index(q_max)


def get_W_status(starts_path, starts_A, weathers):
    s = 1 # 起点
    w = 0 # 背包负重
    w_chunzhuang = 0 # 村庄买的东西
    plan = [W_UP, 0] # 计划购买栈
    Rs = []
    flag = False  # 是否饿死
    for t in range(len(starts_path)-1):# 第0天就开始动，第30天没有动
        where = starts_path[t]
        s_p = starts_A[t] if starts_A[t] != wakuang else where#下一步要干嘛

        w_i, r_i = get_w_i(weathers[t]) # 第t天的基础消耗
        R = -r_i # 基础消耗
        w_i_i = w_i # 基础消耗
        if where!=s_p: # 走，2倍
            R = - 2 * r_i
            w_i_i = 2 * w_i
        if s_p == wakuang: # 挖矿钱
            R = -3 * r_i + shouyi
            w_i_i = 3 * w_i
        w = w + w_i_i
        w_chunzhuang = w_chunzhuang + w_i_i

        if plan[0] >= w_i_i: # 不需要村庄
            plan[0] = plan[0] - w_i_i
        else: # 需要用村庄
            plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
            plan[0] = 0
            if where!=s_p: # 走，2倍
                R = - 4 * r_i
            if s_p == wakuang: # 挖矿钱
                R = -6 * r_i + shouyi # 挖矿钱
            if plan[1] < 0:
                flag = True # 死了

        print("第{}天执行{}->{}".format(t,where,s_p), plan, w_chunzhuang,w)

        # 村庄买东西
        if where in cunzhuang: # 村庄
            plan[1] = plan[1] + w_chunzhuang
            w_chunzhuang = 0

        Rs.append(R)
    return plan,Rs,flag

```

```python

def get_path(mazes, path_map, weathers):
    path = starts_path.copy()
    A = starts_A.copy()
    s = SSSSS
    for t in range(start_t, len(weathers)-1): # (10, 30)
        map = mazes[t,:,:]
        if s==wakuang - 1:
            break
        a = probability_fun(path_map[s], map[s,:], 0)
        s_p = a if a!=wakuang else s

        path.append(s_p)
        A.append(a)
        s = s_p
    return path, A


def get_x_y(weather):
    if weather==1:
        x = 3
        y = 4
    elif weather==2:
        x = 9
        y = 9
    else:
        x = 10
        y = 10
    return x, y


def get_R_x_ys(path,weather,sss):
    zhuan = 0
    for i in path:
        if i==sss:
            zhuan = zhuan+1000
    xs,ys=[],[]
    s=1
    for t in range(1,len(path)):
        a,b = get_x_y(weather[t-1])
        s_p = path[t]
        # s_p=get_index(s,a_pp)
        A=a
        B=b
        if s_p != s and s_p!=sss: # 走，2倍
            A=2*a
            B=2*b
        if s_p == sss:
            A = 3 * a
            B = 3 * b
        xs.append(A)
        ys.append(B)
        # print(s,"->>",s_p," ", A,B)
        s = s_p

    x=sum(xs)
    y=sum(ys)
    R = int((1200 - 2 * y)/3) * 5 + (x + y - int((1200 - 2 * y)/3)) * 10 - zhuan
    return -R,xs,ys,zhuan


def get_awser(maze,path, weather):
    s = 1 # 起点
    w = 0 # 背包负重
    w_chunzhuang = 0 # 村庄买的东西
    plan = [W_UP, 0] # 计划购买栈
    Rs = []
    flag = False
    for t in range(1,len(path)):
        w_i, r_i = get_w_i(weather[t-1])
        R = -r_i # 走一步消耗
        w_i_i = w_i
        if path[t] != s: # 走，2倍
            R = - 2 * r_i
            w_i_i = 2 * w_i
        if path[t] == maze.shape[1]-1:
            R = -3 * r_i + shouyi # 挖矿钱
            w_i_i = 3 * w_i
        w = w + w_i_i
        w_chunzhuang = w_chunzhuang + w_i_i

        if plan[0] >= w_i_i: # 不需要村庄
            plan[0] = plan[0] - w_i_i
            pass
        else: # 需要用村庄
            plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
            plan[0] = 0
            if path[t] != s: # 走，2倍
                R = - 4 * r_i
            if path[t] == maze.shape[1]-1:
                R = -6 * r_i + shouyi # 挖矿钱
        if plan[1] < 0:
            flag = True
        # print(t,s,plan, w_chunzhuang,w)

        # 村庄买东西
        if s in cunzhuang: # 村庄
            plan[1] = plan[1] + w_chunzhuang
            w_chunzhuang = 0

        s = path[t]
        Rs.append(R)
    return plan,Rs,flag

```

```python

def main(gaowengailv, ISREAD_MAZES, PATH):
    mazes = []    # 时间图Q
    weathers = get_weather(gaowengailv)
    for i in range(len(weathers)):
        mazes.append(maze)
    mazes = np.array(mazes).copy()
    path_map = get_pathmap(maze)
    # print(mazes.shape) # mazes (时间, 状态, 动作)
    # print(path_map)
    # print(path_map[21])

    #  mazes付初值
    if ISREAD_MAZES:
        mazes = np.array(np.load("./data/{}_{}.npz".format(PATH,guanqia))['data'])
    # 初值的可视化
    #    cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True, dark=0.7, light=0.3)
    #    sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True, cmap=cmap) #
    #    plt.show()
    mazes_demo = mazes.copy()

    print('######## Q-learing ########')
    plansss, Rs, flag = get_W_status(starts_path, starts_A, weathers)
    # print(flag, plansss)
    print("起点", SSSSS, "第几天", start_t, len(weathers)-1,"矿位置", wakuang)

    huos = 0
    jiage = []
    for k in range(learn_num):# 迭代次数
        s = SSSSS # 起点

        plan = plansss.copy() # 计划购买栈
        path = starts_path.copy()
        w = 1200-plansss[0] # 背包负重
        w_chunzhuang = 1200-plansss[0] # 村庄买的东西
        A = starts_A.copy()
        flag = False
        huo = 0
        # print("*"*50)
        # is_kuang = False
        # print(start_t, len(weathers))
        if ISREAD_MAZES:
            mazes = mazes_demo.copy()
            weathers = get_weather(gaowengailv).copy()

        for t in range(start_t, len(weathers)-1):#(10, 30)

            a = probability_fun(path_map[s], mazes[t, s, :], k=k) # 动作
            # if s not in kuang and a in kuang:
            #    is_kuang=True
            # if is_kuang:
            #    a=s
            #    is_kuang = False

            s_p = a if a != wakuang else s # 下一步要干嘛

            w_i, r_i = get_w_i(weathers[t])
            R = -r_i # 走一步消耗
            w_i_i = w_i
            if a != s: # 走，2倍
                R = - 2 * r_i
                w_i_i = 2 * w_i
            if a == wakuang:
                R = -3 * r_i + shouyi # 挖矿钱
                w_i_i = 3 * w_i
            if weathers[t] == 3: # 沙包必须停留
                s_p = s
                # if a!=wakuang and a!=s:
                #    a=s

            w = w + w_i_i
            w_chunzhuang = w_chunzhuang + w_i_i

            if plan[0] >= w_i_i: # 不需要村庄
                plan[0] = plan[0] - w_i_i
            else: # 需要用村庄
                plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
                plan[0] = 0
                if s != s_p: # 走，2倍
                    R = - 4 * r_i
                if a == wakuang: # 挖矿钱
                    R = -6 * r_i + shouyi # 挖矿钱
                if plan[1] < 0:
                    flag = True # 死了
            if plan[1] < 0:
                flag = True # 死了

            # print("第{}天执行{}，{}->{}".format(t,a,s,s_p), plan)

            # 村庄买东西
            if s in cunzhuang: # 村庄
                plan[1] = plan[1] + w_chunzhuang
                w_chunzhuang = 0

            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

            if s_p == wakuang - 1: # 终点
                huo = 1
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
                s = s_p # 更新位置
```

```
                # print(path)
                break      # 29
            elif flag: # 中止条件
                R = R - 1000000
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
                    + gamma * max_q)
                s = s_p # 更新位置
                break
            elif t == len(weathers) - 1 -1 and s_p != wakuang-1:
                R = R - 1000000
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
                    + gamma * max_q)
                s = s_p # 更新位置
                break
            else:
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
                    + gamma * max_q)
                s = s_p # 更新位置
            A.append(a)
            path.append(s)
        # break


        A.append(25)
        pathsss = [1]
        for i in A:
            pathsss.append(i)
        if huo:
            huos = huos + 1 # 活着
            R, xs, ys, zhuan = get_R_x_ys(pathsss, weathers, 26)
            # print(R)
            # print(pathsss)
            jiage.append(R)

    if ISREAD_MAZES:
        assert len(jiage)==huos
        print("平均获得资金", sum(jiage)/huos, '活着概率',round(huos*100/learn_num,2),"%")
    else:
        np.savez_compressed("./data/{}_{}.npz".format(PATH,guanqia),data=mazes)

        path, A = get_path(mazes, path_map, weathers)
        # print(path)
        # print(A)
        paths = [1]
        for i in A:
            paths.append(i)

        # print(len(paths))
        plan,Rs, flag = get_awser(maze, paths, weathers)
        print(paths)
        # print(plan)
        # print(Rs)
        print("****************************")
        print(flag)
        R, xs, ys, zhuan = get_R_x_ys(paths, weathers, 26)
        print(R)


plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签


if __name__ == '__main__':

    for gaowengailv in np.arange(0,1-shabao,0.1):
        ISREAD_MAZES = False
        yuzhi = int(learn_num * 0.9) if not ISREAD_MAZES else 0 # 策略域

        gaowengailv2 = gaowengailv+shabao
        main(gaowengailv2, ISREAD_MAZES,
            'demo4_gailv_1_01_{}'.format(int(gaowengailv * 100)))

        ISREAD_MAZES = True
        yuzhi = int(learn_num * 0.9) if not ISREAD_MAZES else 0 # 策略域
        main(gaowengailv2, ISREAD_MAZES,
            'demo4_gailv_1_01_{}'.format(int(gaowengailv * 100)))


    huozhe = [ 78.76,83.31 , 74.14, 97.38 , 86.76,
               75.12,67.75, 61.09, 41.09, 34.54]
    jiage = [14165,11166.5,8238.2,3658.34,6177.5,
             2303.523,1376.023,702.071,103.6772,-311.41]

    huozhe2 = [63.39,33.8 , 26.38, 95.61 , 86.76,
               45.12,37.75, 21.09, 15.88, 10.19]
    jiage2 = [14565,11176.962,7463.11,4256.00, 7125.6496,
             5160.709,3305.03,824.4983,315.22,-281.64]

    x = np.arange(0,0.95,0.1)+shabao
    fig, ax1 = plt.subplots()
    ax1.plot(x, huozhe, label="保守策略时存活率", linestyle='--',
       marker='*')
    ax1.plot(x, huozhe2, label="非保守策略时存活率", linestyle='--',
       marker='*')
    ax1.plot([1], [0], linestyle=':')
    plt.grid(linestyle='-.')
    ax1.set_xlabel("高温概率")
    ax1.set_ylabel("存活率/%")
    fig.legend(loc=1, bbox_to_anchor=(1,1), bbox_transform=ax1.transAxes)
    plt.show()

    fig, ax2 = plt.subplots() # 做镜像处理
    ax2.plot(x, jiage, label="保守策略时平均保留资金")
    ax2.plot(x, jiage2, '-r',label="非保守策略时平均保留资金")
```

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

```
import numpy as np
from maze import get_maze
import random
import matplotlib.pyplot as plt
import seaborn as sns
guanqia = 5 #关卡
learn_num = 10000 #迭代次数
yuzhi = int(learn_num*0.9) # 策略域
W_UP = 1200  #负重上线
aleph, gamma = 1, 0.8 #学习率
shouyi = 200 # 挖矿收益
cunzhuang = [15]
ISREAD_MAZES = False


def get_w_i(weather):
    w_i, p_i = 0, 0
    if weather==1:
        w_i = 3*5+2*7
        p_i = 5*5+10*7
    elif weather==2:
        w_i = 3*8+2*6
        p_i = 5*8+10*6
    else:
        w_i = 3*10+2*10
        p_i = 5*10+10*10
    return w_i, p_i


def probability_fun(sss, max_a, probability=0.8, k=None):
    '''
    :param probability: 0为贪婪，1为随机策略，其他为sigema策略
    '''
    if probability==1:
        return random.choice(sss)
    elif probability==0:
        list_a = max_a.tolist()
        q_max = float('-inf')
        for i in sss:
            if list_a[i] > q_max and not np.isnan(list_a[i]):
                q_max=list_a[i]
        # if probability==0:
        #     print(q_max, list_a.index(q_max))
        return list_a.index(q_max)
    else:
        if k<yuzhi:
            probability=-(1-0.5)*k/yuzhi+1
        else:
            probability=0
        if np.random.random()<probability:
            return random.choice(sss)
        else:
            list_a = max_a.tolist()
            q_max = float('-inf')
            for i in sss:
                if list_a[i] > q_max and not np.isnan(list_a[i]):
                    q_max = list_a[i]
            # if probability==0:
```

```python
            #    print(q_max, list_a.index(q_max))
            return list_a.index(q_max)


def get_index(s,a,map):
    if a!=map.shape[1]-1:
        return a
    else:
        return s


def get_path(mazes, weather, path_map):
    s=1
    path = []
    path.append(s)
    for t in range(len(weather)):
        map = mazes[t,:,:]
        if s==map.shape[0]-1:
            break
        a = probability_fun(path_map[s], map[s,:], 0)
        path.append(a)
        s_p = get_index(s, a, map)
        s = s_p
    return path


def get_pathmap(maze):
    path_map = []
    #每个状态拥有的动作
    for i in range(maze.shape[0] ): # 初始化所有可行域
        temp = []
        for j in range(1, maze.shape[1]):
            if not np.isnan(maze[i,j]):
                temp.append(j)
        path_map.append(temp)
    return path_map


def init_mazes(mazes, path, weather):
    for k in range(10000):
        # 判断终止条件
        print("*" * 50)
        s = 1 # 起点
        w = 0 # 背包负重
        w_chunzhuang = 0 # 村庄买的东西
        plan = [W_UP, 0] # 计划购买栈
        flag = False

        for t in range(1, len(path)):

            a = path[t]
            s_p=get_index(s, a, mazes[t,:,:]) # 动作执行完后状态

            w_i, r_i = get_w_i(weather[t])
            R = -r_i # 走一步消耗
            w_i_i = w_i
            if a != s: # 走，2倍
                R = - 2 * r_i
                w_i_i = 2 * w_i
            if a == maze.shape[1] - 1:
                R = -3 * r_i + shouyi # 挖矿钱
                w_i_i = 3 * w_i

            w = w + w_i_i
            w_chunzhuang = w_chunzhuang + w_i_i

            ##################
            # 村庄买东西
            if plan[0] >= w_i_i: # 不需要村庄
                plan[0] = plan[0] - w_i_i
                pass
            else: # 需要用村庄
                plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
                plan[0] = 0
                if path[t] != s: # 走，2倍
                    R = - 4 * r_i
                if path[t] == maze.shape[1] - 1:
                    R = -6 * r_i + shouyi # 挖矿钱
                if plan[1] < 0:
                    flag = True

            # 村庄买东西
            if s in cunzhuang: # 村庄
                plan[1] = plan[1] + w_chunzhuang
                w_chunzhuang = 0
            ##################
            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
            if 15 in path:
                print(path, max_q, plan, w, w_chunzhuang, R, )

            if s_p == lenth - 1: # 终点
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
                s = s_p # 更新位置
                break
            elif flag or (t == len(weather) - 1 - 1 and s_p != lenth - 1): # 中止条件
                R = R - 1000000
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
                s = s_p # 更新位置
                break
            else:
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
```

```python
                s = s_p # 更新位置
    np.savez_compressed("./data/guan_init{}.npz".format(guanqia), data=mazes)
    return mazes


def get_awser(maze,path, weather):
    s = 1 # 起点
    w = 0 # 背包负重
    w_chunzhuang = 0 # 村庄买的东西
    plan = [W_UP, 0] # 计划购买栈
    Rs = []
    flag = False
    for t in range(1,len(path)):
        w_i, r_i = get_w_i(weather[t-1])
        R = -r_i # 走一步消耗
        w_i_i = w_i
        if path[t] != s: # 走, 2倍
            R = - 2 * r_i
            w_i_i = 2 * w_i
        if path[t] == maze.shape[1]-1:
            R = -3 * r_i + shouyi # 挖矿钱
            w_i_i = 3 * w_i
        w = w + w_i_i
        w_chunzhuang = w_chunzhuang + w_i_i

        if plan[0] >= w_i_i: # 不需要村庄
            plan[0] = plan[0] - w_i_i
            pass
        else: # 需要用村庄
            plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
            plan[0] = 0
            if path[t] != s: # 走, 2倍
                R = - 4 * r_i
            if path[t] == maze.shape[1]-1:
                R = -6 * r_i + shouyi # 挖矿钱
        if plan[1] < 0:
            flag = True
        print(t,s,plan, w_chunzhuang,w)

        # 村庄买东西
        if s in cunzhuang: # 村庄
            plan[1] = plan[1] + w_chunzhuang
            w_chunzhuang = 0

        s = path[t]
        Rs.append(R)
    return plan,Rs,flag


from get_R import get_R_x_y

if __name__ == '__main__':
    maze, weather = get_maze(guanqia)
    path = [1,4,6,13]
    print(len(path))
    plan,Rs, flag = get_awser(maze, path, weather)
    print(path)
    print(plan)
    print(Rs)
    print("****************************")
    print(flag)
    R,xs,ys = get_R_x_y(path, weather, 28)
    print(R)

if __name__ == '__main__ssss;':
    maze, weather = get_maze(guanqia)
    lenth = maze.shape[0] #终点位置
    print(weather, lenth)
    mazes = []    # 时间图
```

```
for i in range(len(weather)):
    mazes.append(maze)
mazes = np.array(mazes)
path_map = get_pathmap(maze)
print(mazes.shape) # mazes (时间, 状态, 动作)
print(path_map)

# R放入初值中
for t in range(len(weather)):
    for j in range(lenth): # 初始化所有可行域
        for a in path_map[j]:
            w_i, r_i = get_w_i(weather[t])
            R = -r_i
            if a != j: # 走, 2倍
                R = - 2 * r_i
            if a == maze.shape[1] - 1:
                R = -3 * r_i + shouyi # 挖矿钱
            mazes[t,j,a] = R

init_path = [1, 25, 24, 23, 23, 22, 9, 9, 15, 14, 12, 28, 28, 28, 28, 28, 12, 28, 28, 13, 15, 9, 21, 27]

cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True, dark=0.7, light=0.3)
sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True, cmap=cmap) #
plt.show()

mazes = init_mazes(mazes,init_path,weather)
print(mazes)
# mazes = np.array(np.load("./data/guan_init{}.npz".format(guanqia))['data'])

if ISREAD_MAZES:
    mazes = np.array(np.load("./data/guan{}.npz".format(guanqia))['data'])
```

```

cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True, dark=0.7, light=0.3)
sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True, cmap=cmap) #
plt.show()


##q-learing
print('######## Q-learing ########')
for k in range(learn_num):# 迭代次数
    # 判断终止条件
    print("*"*50)
    s = 1 # 起点
    w = 0 # 背包负重
    w_chunzhuang = 0  # 村庄买的东西
    plan = [W_UP, 0] # 计划购买栈
    path = [s]
    flag = False

    for t in range(len(weather)-1):
        a = probability_fun(path_map[s], mazes[t,s,:], k=k) #动作
        s_p = get_index(s, a, mazes[t,:,:]) # 动作执行完后状态
        w_i, r_i = get_w_i(weather[t])
        R = -r_i             # 走一步消耗
        w_i_i = w_i
        if a != s:  # 走, 2倍
            R = - 2 * r_i
            w_i_i = 2*w_i
        if a==maze.shape[1]-1:
            R = -3*r_i+shouyi     # 挖矿钱
            w_i_i = 3*w_i
        if weather[t]==3: # 沙包必须停留
            s_p = s

        w = w + w_i_i
```

```
            w_chunzhuang = w_chunzhuang + w_i_i

        ##################
        # 村庄买东西
        if plan[0] >= w_i_i: # 不需要村庄
            plan[0] = plan[0] - w_i_i
            pass
        else: # 需要用村庄
            plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
            plan[0] = 0
            if path[t] != s: # 走，2倍
                R = - 4 * r_i
            if path[t] == maze.shape[1] - 1:
                R = -6 * r_i + shouyi # 挖矿钱
            if plan[1] < 0:
                flag = True

        # 村庄买东西
        if s in cunzhuang: # 村庄
            plan[1] = plan[1] + w_chunzhuang
            w_chunzhuang = 0
        ##################

        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
        if 15 in path:
            print(path, max_q, plan, w, w_chunzhuang, R,)

        if s_p==lenth-1:# 终点
            mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
            s = s_p # 更新位置
            break
        elif flag or (t==len(weather)-1-1 and s_p != lenth-1):#中止条件
            R = R - 1000000
            mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
                 + gamma*max_q)
            s = s_p # 更新位置
            break
        else:
            mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R
                 + gamma*max_q)
            s = s_p # 更新位置
        # print(mazes[t,s,a])
        path.append(a)
    pass


# 输出路径
# print(mazes)
np.savez_compressed("./data/guan{}.npz".format(guanqia), data=mazes)

path = get_path(mazes, weather, path_map)
print(path)

#
# for t in range(25, len(weather)):
#    map = mazes[t,:,:]
#    for i in range(map.shape[0]):
#        for j in range(map.shape[1]):
#            if map[i,j]<-100000 and not np.isnan(map[i,j]):
#                map[i, j] = -1000000
#    cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True,
   dark=0.7, light=0.3)
#    sns.heatmap(map, linewidths=0.05, cbar=True, cmap=cmap) #
#    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-learing 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。

```python
import numpy as np
from maze import get_maze
import random
import matplotlib.pyplot as plt
import seaborn as sns

jilu_jiage = []

# ISREAD_MAZES = True

guanqia = 6 #关卡
learn_num = 100000 #迭代次数
W_UP = 1200  #负重上线
aleph, gamma = 0.8, 0.8 #学习率
shouyi = 1000 # 挖矿收益
ddd = 30   # 天数
cunzhuang = [14]
kuang = [18]      # 矿山
maze, _ = get_maze(guanqia)
wakuang = maze.shape[0] # 终点位置
# PATH = 'demo4'
starts_path= [1]
starts_A= []
start_t = len(starts_path) - 1
SSSSS = starts_path[len(starts_path) - 1]# 起点
# gaowengailv = 0.3 + 0.05 # 高温概率(其中0.05是沙暴概率，固定)
shabao = 0.05


def get_weather(gaowengailv):
    weatherssss = []
    for i in range(ddd):
        x = random.random()
        if x<shabao:
            weatherssss.append(3)
        elif x>=shabao and x<gaowengailv:
            weatherssss.append(2)
        else:
            weatherssss.append(1)
    return weatherssss


def get_w_i(weather):
    w_i, p_i = 0, 0
    if weather==1:
        w_i = 3*3+2*4
        p_i = 3*5+10*4
    elif weather==2:
        w_i = 9*3+2*9
        p_i = 9*5+10*9
    else:
        w_i = 3*10+2*10
        p_i = 5*10+10*10
    return w_i, p_i


def get_pathmap(maze):
    path_map = []
    # 每个状态拥有的动作
    for i in range(maze.shape[0]): # 初始化所有可行域
        temp = []
        for j in range(1, maze.shape[1]):
            if not np.isnan(maze[i,j]):
                temp.append(j)
        path_map.append(temp)
    return path_map


def probability_fun(sss, max_a, probability=0.8, k=None):
    '''
    :param probability: 0为贪婪，1为随机策略，其他为sigema策略
    '''
    if probability==1:
        return random.choice(sss)
    elif probability==0:
        list_a = max_a.tolist()
        q_max = float('-inf')
        for i in sss:
            if list_a[i] > q_max and not np.isnan(list_a[i]):
                q_max=list_a[i]
        # if probability==0:
        #     print(q_max, list_a.index(q_max))
        return list_a.index(q_max)
    else:
        if k<yuzhi:
            probability=-(1-0.5)*k/yuzhi+1
        else:
            probability=0
        if np.random.random()<probability:
            return random.choice(sss)
        else:
            list_a = max_a.tolist()
            q_max = float('-inf')
            for i in sss:
                if list_a[i] > q_max and not np.isnan(list_a[i]):
                    q_max = list_a[i]
            # if probability==0:
            #     print(q_max, list_a.index(q_max))
            return list_a.index(q_max)


def get_W_status(starts_path, starts_A, weathers):
    s = 1 # 起点
    w = 0 # 背包负重
    w_chunzhuang = 0 # 村庄买的东西
    plan = [W_UP, 0] # 计划购买栈
    Rs = []
    flag = False  # 是否饿死
    for t in range(len(starts_path)-1):# 第0天就开始动，第30天没有动
        where = starts_path[t]
        s_p = starts_A[t] if starts_A[t] != wakuang else where#下一步要干嘛

        w_i, r_i = get_w_i(weathers[t]) # 第t天的基础消耗
        R = -r_i # 基础消耗
        w_i_i = w_i # 基础消耗
        if where!=s_p: # 走，2倍
            R = - 2 * r_i
            w_i_i = 2 * w_i
        if s_p == wakuang: # 挖矿钱
            R = -3 * r_i + shouyi
            w_i_i = 3 * w_i
        w = w + w_i_i
        w_chunzhuang = w_chunzhuang + w_i_i

        if plan[0] >= w_i_i: # 不需要村庄
            plan[0] = plan[0] - w_i_i
        else: # 需要用村庄
            plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
            plan[0] = 0
            if where!=s_p: # 走，2倍
                R = - 4 * r_i
            if s_p == wakuang: # 挖矿钱
                R = -6 * r_i + shouyi # 挖矿钱
            if plan[1] < 0:
                flag = True # 死了

        print("第{}天执行{}->{}".format(t,where,s_p), plan, w_chunzhuang,w)
```

```python

        # 村庄买东西
        if where in cunzhuang: # 村庄
            plan[1] = plan[1] + w_chunzhuang
            w_chunzhuang = 0

        Rs.append(R)
    return plan,Rs,flag


def get_path(mazes, path_map, weathers):
    path = starts_path.copy()
    A = starts_A.copy()
    s = SSSSS
    for t in range(start_t, len(weathers)-1): # (10, 30)
        map = mazes[t,:,:]
        if s==wakuang - 1:
            break
        a = probability_fun(path_map[s], map[s,:], 0)
        s_p = a if a!=wakuang else s

        path.append(s_p)
        A.append(a)
        s = s_p
    return path, A


def get_x_y(weather):
    if weather==1:
        x = 3
        y = 4
    elif weather==2:
        x = 9
        y = 9
    else:
        x = 10
        y = 10
    return x, y


def get_R_x_ys(path,weather,sss):
    zhuan = 0
    for i in path:
        if i==sss:
            zhuan = zhuan+1000
    xs,ys=[],[]
    s=1
    for t in range(1,len(path)):
        a,b = get_x_y(weather[t-1])
        s_p = path[t]
        # s_p=get_index(s,a_pp)
        A=a
        B=b
        if s_p != s and s_p!=sss: # 走, 2倍
            A=2*a
            B=2*b
        if s_p == sss:
            A = 3 * a
            B = 3 * b
        xs.append(A)
        ys.append(B)
        # print(s,"->>",s_p," ", A,B)
        s = s_p

    x=sum(xs)
    y=sum(ys)
    R = int((1200 - 2 * y)/3) * 5 + (x + y - int((1200 - 2 * y)/3)) * 10
        - zhuan
    return -R,xs,ys,zhuan

```

```python

def get_awser(maze,path, weather):
    s = 1 # 起点
    w = 0 # 背包负重
    w_chunzhuang = 0 # 村庄买的东西
    plan = [W_UP, 0] # 计划购买栈
    Rs = []
    flag = False
    for t in range(1,len(path)):
        w_i, r_i = get_w_i(weather[t-1])
        R = -r_i # 走一步消耗
        w_i_i = w_i
        if path[t] != s: # 走, 2倍
            R = - 2 * r_i
            w_i_i = 2 * w_i
        if path[t] == maze.shape[1]-1:
            R = -3 * r_i + shouyi # 挖矿钱
            w_i_i = 3 * w_i
        w = w + w_i_i
        w_chunzhuang = w_chunzhuang + w_i_i

        if plan[0] >= w_i_i: # 不需要村庄
            plan[0] = plan[0] - w_i_i
            pass
        else: # 需要用村庄
            plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
            plan[0] = 0
            if path[t] != s: # 走, 2倍
                R = - 4 * r_i
            if path[t] == maze.shape[1]-1:
                R = -6 * r_i + shouyi # 挖矿钱
        if plan[1] < 0:
            flag = True
        # print(t,s,plan, w_chunzhuang,w)

        # 村庄买东西
        if s in cunzhuang: # 村庄
            plan[1] = plan[1] + w_chunzhuang
            w_chunzhuang = 0

        s = path[t]
        Rs.append(R)
    return plan,Rs,flag


def main(gaowengailv, ISREAD_MAZES, PATH):
    mazes = []     # 时间图Q
    weathers = get_weather(gaowengailv)
    for i in range(len(weathers)):
        mazes.append(maze)
    mazes = np.array(mazes).copy()
    path_map = get_pathmap(maze)
    # print(mazes.shape) # mazes (时间, 状态, 动作)
    # print(path_map)
    # print(path_map[21])

    #  mazes付初值
    if ISREAD_MAZES:
        mazes = np.array(np.load("./data/{}_{}.npz".format(PATH,guanqia))['data'])
    # 初值的可视化
    #    cmap = sns.cubehelix_palette(start=3, rot=4, as_cmap=True, dark=0.7, light=0.3)
    #    sns.heatmap(mazes[0,:,:], linewidths=0.05, cbar=True, cmap=cmap) #
    #    plt.show()
    mazes_demo = mazes.copy()

    print('######## Q-learing ########')
    plansss, Rs, flag = get_W_status(starts_path, starts_A, weathers)
```

```python
    # print(flag, plansss)
    print("起点", SSSSS, "第几天", start_t, len(weathers)-1,"矿位置", wakuang)

    huos = 0
    jiage = []
    for k in range(learn_num):# 迭代次数
        s = SSSSS # 起点

        plan = plansss.copy() # 计划购买栈
        path = starts_path.copy()
        w = 1200-plansss[0] # 背包负重
        w_chunzhuang = 1200-plansss[0] # 村庄买的东西
        A = starts_A.copy()
        flag = False
        huo = 0
        # print("*"*50)
        # is_kuang = False
        # print(start_t, len(weathers))
        if ISREAD_MAZES:
            mazes = mazes_demo.copy()
            weathers = get_weather(gaowengailv).copy()

        for t in range(start_t, len(weathers)-1):#(10, 30)

            a = probability_fun(path_map[s], mazes[t, s, :], k=k) # 动作
            # if s not in kuang and a in kuang:
            #     is_kuang=True
            # if is_kuang:
            #     a=s
            #     is_kuang = False

            s_p = a if a != wakuang else s # 下一步要干嘛

            w_i, r_i = get_w_i(weathers[t])
            R = -r_i # 走一步消耗
            w_i_i = w_i
            if a != s: # 走，2倍
                R = - 2 * r_i
                w_i_i = 2 * w_i
            if a == wakuang:
                R = -3 * r_i + shouyi # 挖矿钱
                w_i_i = 3 * w_i
            if weathers[t] == 3: # 沙包必须停留
                s_p = s
                # if a!=wakuang and a!=s:
                #    a=s

            w = w + w_i_i
            w_chunzhuang = w_chunzhuang + w_i_i

            if plan[0] >= w_i_i: # 不需要村庄
                plan[0] = plan[0] - w_i_i
            else: # 需要用村庄
                plan[1] = plan[1] - w_i_i + plan[0] # 村庄装不够的
                plan[0] = 0
                if s != s_p: # 走，2倍
                    R = - 4 * r_i
                if a == wakuang: # 挖矿钱
                    R = -6 * r_i + shouyi # 挖矿钱
                if plan[1] < 0:
                    flag = True # 死了
            if plan[1] < 0:
                flag = True # 死了

            # print("第{}天执行{}，{}->{}".format(t,a,s,s_p)，plan)

            # 村庄买东西
            if s in cunzhuang: # 村庄
                plan[1] = plan[1] + w_chunzhuang
                w_chunzhuang = 0

            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

            if s_p == wakuang - 1: # 终点
                huo = 1
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * R
                s = s_p # 更新位置
                # print(path)
                break      # 29
            elif flag: # 中止条件
                R = R - 1000000
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
                s = s_p # 更新位置
                break
            elif t == len(weathers) - 1 -1 and s_p != wakuang-1:
                R = R - 1000000
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
                s = s_p # 更新位置
                break
            else:
                mazes[t, s, a] = (1 - aleph) * mazes[t, s, a] + aleph * (R + gamma * max_q)
                s = s_p # 更新位置
            A.append(a)
            path.append(s)
        # break


        A.append(25)
        pathsss = [1]
        for i in A:
            pathsss.append(i)
        if huo:
            huos = huos + 1 # 活着
            R, xs, ys, zhuan = get_R_x_ys(pathsss, weathers, 26)
            # print(R)
            # print(pathsss)
            jiage.append(R)

    if ISREAD_MAZES:
        assert len(jiage)==huos
        print("平均收益", sum(jiage)/huos, '活着概率', round(huos*100/learn_num,2),"%")
    else:
        np.savez_compressed("./data/{}_{}.npz".format(PATH,guanqia), data=mazes)

        path, A = get_path(mazes, path_map, weathers)
        # print(path)
        # print(A)
        paths = [1]
        for i in A:
            paths.append(i)

        # print(len(paths))
        plan,Rs, flag = get_awser(maze, paths, weathers)
        print(paths)
        # print(plan)
        # print(Rs)
        print("****************************")
        print(flag)
        R, xs, ys, zhuan = get_R_x_ys(paths, weathers, 26)
        print(R)


plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
```

```python


if __name__ == '__main__':

    # for gaowengailv in np.arange(0,1-shabao,0.1):
    #    ISREAD_MAZES = False
    #    yuzhi = int(learn_num * 0.9) if not ISREAD_MAZES else 0 # 策略域
    #
    #    gaowengailv2 = gaowengailv+shabao
    #    main(gaowengailv2, ISREAD_MAZES, 'demo6_gailv_2_01_{}'.format(int(gaowengailv * 100)))
    #
    #    ISREAD_MAZES = True
    #    yuzhi = int(learn_num * 0.9) if not ISREAD_MAZES else 0 # 策略域
    #    main(gaowengailv2, ISREAD_MAZES, 'demo6_gailv_2_01_{}'.format(int(gaowengailv * 100)))


    huozhe = np.array([88.05,61.2,25.03,64.89,34.8,
             30.23,80.05 ,12.1 ,21.5, 87.07])
    jiage = np.array([10461.8539,13351.1500,9176.8173,5457.4128,5252.251,
           4217.9555, 1592.151,3037.4466,399.0921,111.672])/3

    huozhe2 = np.array([64.15,25.89,4.66,44.96,57.35 ,
              29.9, 35.93,38.49,96.08, 14.27,])
    jiage2 =np.array( [12210.633,8490.3822,11078.8149, 6299.4749, 4370.22,
            4201.05424,3225.40552,2148.689,-208.6307,407.588972])/3

    huozhe3 = np.array([26.29 , 91.12 ,24.6,8.31,6.87 ,
              88.73, 35.9,38.0,50.37, 87.23,])
    jiage3 = np.array([14165.0 ,2747.553, 9173.948, 6299.4749, 7124.87,
            2045.927, 3196.399,2165.566, 1098.84858,100.521])/3

    huozhe = (huozhe+huozhe2+huozhe3)/3
```

```
x = np.arange(0,0.95,0.1)+shabao
plt.plot(x, huozhe, label="平均存活率", linestyle='--', marker='*')
plt.ylim((0,100))
plt.show()



x = np.arange(0,0.95,0.1)+shabao
fig, ax1 = plt.subplots()
ax1.plot(x, huozhe, label="agent1存活率", linestyle='--', marker='*')
ax1.plot(x, huozhe2, label="agent2存活率", linestyle='--', marker='*')
ax1.plot(x, huozhe3, label="agent3存活率", linestyle='--', marker='*')
ax1.plot([1], [0], linestyle=':')
plt.grid(linestyle='-.')
ax1.set_xlabel("高温概率")
ax1.set_ylabel("存活率/%")
fig.legend(loc=1, bbox_to_anchor=(1,1), bbox_transform=ax1.transAxes)
plt.show()

fig, ax2 = plt.subplots() # 做镜像处理
ax2.plot(x, jiage, label="agent1平均保留资金")
ax2.plot(x, jiage2, '-r',label="agent2平均保留资金")
ax2.plot(x, jiage3, '-k',label="agent3平均保留资金")
plt.grid(linestyle='-.')
ax2.set_ylabel("平均保留资金")
ax2.set_xlabel("高温概率")
fig.legend(loc=1, bbox_to_anchor=(1,1), bbox_transform=ax1.transAxes)
# plt.savefig(r'C:\Users\77526\Desktop\CUMCM\precode\pic\iteration.png')
plt.show()
```