基于风险决策和非合作博弈的穿越沙漠策略设计

摘要

本文首先对不同天气信息情报的情况分别建立了**优化模型**和**风险决策模型**，并通过深度优先搜索算法得到了相应的策略，并采用蒙特卡洛模拟验证策略的有效性。另外，针对存在多人博弈的情况，建立了**非合作博弈模型**，并用**帝国竞争算法**求解满足纳什均衡的混合策略。

针对问题一，本文考虑资源消耗约束、负载约束、区域相邻约束和起止点约束，建立以最大化剩余资金作为目标的**优化模型**。在求解的过程中，引入事件触发点的概念，并通过 **Floyd 算法**确定了这类节点之间的最短路径，将问题转化为求解事件触发点的最优组合方案。进而基于约束条件设计**剪枝策略**，结合**深度优先搜索算法**求解得到了最佳行动方案。第一关最大剩余资金为 **10470 元**；第二关则为 **12355 元**。

针对问题二，考虑到玩家仅能知道当天的天气情况，本文建立了**风险决策模型**分析最佳策略。首先根据**马尔可夫决策过程**，综合考虑短期行动资源消耗和长期资金获取和存活的目标构建了**回报函数**，并基于**最大期望决策收益准则**确定各状态下的行动策略。为验证模型的合理性，本文采用**蒙特卡洛算法**模拟出天气序列，并基于上述策略进行仿真。对于第三关，玩家的通关率为 **99.98%**，剩余资金期望为 **9377.4 元**；对于第四关，玩家的通关率为 **88.675%**，平均剩余资金为 **8322.5 元**。通过灵敏度分析发现，沙暴概率的增加对玩家通过率的影响较小，策略较为保守。

针对问题三，考虑天气全部已知和仅知道当前天气两种情况分别建立**博弈模型**。对于第五关，沿用问题一中事件触发点和有效路径的概念，确定了两名玩家的策略集，之后通过计算**双人非合作博弈模型**得到两者的混合策略均为 **(0.5,0.5)**，期望支付为 **752.5 元**。对于第六关，本文采用问题二中的**回报函数**，得到了任一状态下各行为策略的**效益期望值**，并结合多人相同行为的惩罚对其进行修正。进而建立**多人多阶段非合作博弈模型**，并通过**帝国竞争算法**求解得到平衡局势。通过分析策略发现，本文制定的策略不仅考虑到了自身的状态，同时还能够根据对手的状态进行有效的决策。

本文的优点包括：1. 通过引入事件触发点的概念，减少了讨论的节点数，使模型得到了简化，增加了求解速度；2. 玩家的行动策略充分考虑到自身的状态以及行为带来的短期及长期的影响，保证通过的同时尽可能使收益最大化。

关键字： **穿越沙漠策略**　**深度优先搜索算法**　**风险决策模型**　**非合作博弈**

1 问题重述

1.1 问题背景

玩家根据地图在沙漠中行走，在起始点可以获取一定量的水和食物，在沙漠的途中会碰到不同的天气。在地图中有矿山和村庄，玩家可以在矿山采矿以补给资金，也可以在村庄补充水和食物。玩家的目标为在规定时间内回到终点，并手头留下尽可能多的资金。

1.2 问题提出

问题一需要基于第一关和第二关给定的限制条件，由一名玩家在未来天气全部已知的情况下通过第一关和第二关，并求出玩家的最优策略。

问题二需要基于第三关和第四关给定的限制条件，由一名玩家在未来天气未知，且只知道当日天气的情况下通过第三关和第四关，并给出一般情况下的玩家的最佳策略。

问题三假设存在 n 名玩家，玩家之间会发生相互影响，分别求得天气已知和未知情况下的每名玩家应采取的策略。

2 模型的假设

- 问题一中玩家在事件触发点之间通行时只会选择固定的最短路线;
- 到达终点时正好消耗掉所有食物和水，不存在半价卖回的情况;
- 各玩家都符合理性人的设定，意味着在此游戏中玩家寻求最终利益最大化。

3 符号说明

符号	意义
$d_{ii'}$	定义区域之间是否相邻的邻接矩阵
Q_{kj}	第 j 天玩家库存的资源量
S_{kj}	第 j 天玩家消耗的资源量
T_{kj}	第 j 天玩家获得的资源量
P	天气转移概率

注：其他未列出的符号以第一次出现时的解释为准。

4 问题一模型建立与求解

4.1 问题分析

观察游戏的规则可以发现，若以挖矿获得的收益为正，则行动消耗的资源为负，且每一天消耗的资源都会因为天气的改变而不同，故可将其视为一个带负权的时变最短路问题。而 Bellman-Ford 等传统的带负权最短路算法并不能处理时变的问题。因此，本文拟将本问题转化为线性规划模型，并将相应的约束作为剪枝策略，采用深度优先搜索算法对问题进行求解。目前已经有学者证明了时变最短路是一个 NPC 问题 [1]，因此在问题求解的过程中，可以考虑对问题进行简化，降低求解难度。

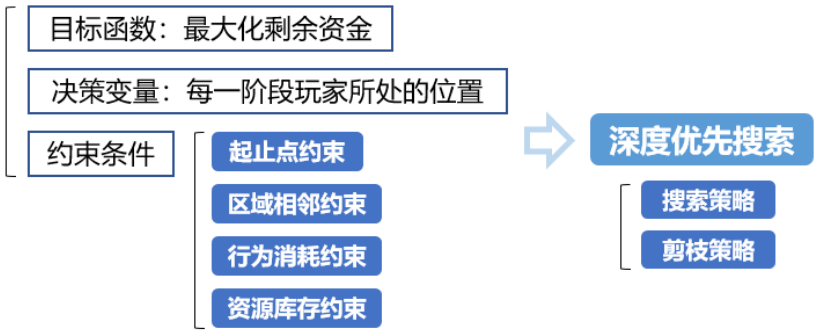


图 1　问题一思路图

4.2 模型建立

根据问题中所给的游戏规则，以及附件中的地图约束建立模型，模型具体如下：

4.2.1 决策变量

玩家的行动序列可以被拆分为具体到每一天的状态，本文以 0-1 变量表示玩家在第 j 天是否在区域 i，用符号 x_{ij} 表示:

$$x_{ij} = \begin{cases} 1, & \text{第 j 天玩家在第 i 区域;} \\ 0, & \text{第 j 天玩家不在第 i 区域.} \end{cases} \tag{1}$$

4.2.2 目标函数

设 W 为到达终点后玩家剩下的金钱（将食物和水也同时按照游戏规则折合为金钱），要使得最后到达终点后玩家剩下的金钱最大化。

$$\max W = \sum_{j} X_{12j} \times 1000 + 10000 - \sum_{k} [(T_{k1} + \sum_{j=2}^{27} 2T_k j)a_k]. \tag{2}$$

4.2.3 约束条件

1) 起止点约束：约束玩家必须从起点出发，到终点结束:

$$\begin{cases} \sum_{j=0}^{30} x_{27j} = 1; \\ \sum_{j=0}^{30} x_{1j} \geq 1; \end{cases} \tag{3}$$

2) 区域相邻约束： 玩家从 j 时刻的 i 区域到 $j+1$ 时刻的 i' 区域，必须满足 i 和 i' 两区域相邻：

$$x_{ij} x_{i'j+1} \leq d_{ii'} \tag{4}$$

其中，$d_{ii'}$ 为 0-1 变量，用以表示两区域是否相邻：

$$d_{ii'} = \begin{cases} 1 & \text{第 i 个区域和第 i' 个区域相邻;} \\ 0 & \text{第 i 个区域和第 i' 个区域不相邻.} \end{cases} \tag{5}$$

3) 行为消耗： 玩家在其他区域的消耗为 $S_{kj} = V_j \times a_{kj}$, 当 $k = 1, 2$ 分别水和食物，S_i 表示第 i 种物资的消耗量。当玩家到达矿山，并选择进行挖矿时，每日的食物和水

消耗是基础的三倍：

$$V_j \leq 2 + Mx_{12j}, \tag{6}$$

其中 V_j 为第 j 天的行为，M 为一个极大的值。

4) 库存约束：若水和食物的库存不能满足下一日的需求，玩家将输掉游戏，故对此进行约束：

$$\begin{cases} Q_{kj} + T_{kj} - S_{kj} \geq 0, \\ Q_{kj+1} = Q_{kj} + T_{kj} - S_{kj}, \\ 3Q_{1j} + 2Q_{2j} \leq 1200, \end{cases} \tag{7}$$

式中，Q_{kj} 为第 j 天一开始水和食物的量，T_{kj} 为第 j 天购买的水和食物量，S_{kj} 为第 j 天物资 k 的消耗。分别表示一天之后玩家剩下的库存一定大于零，且玩家剩下的库存为第二天的起始库存, 并满足库存总重量小于最大载重量。

4.2.4 模型总述

以下为玩家到达终点后剩下金钱最大化的规划模型：

$$\max \sum_j X_{12j} \times 1000 + 10000 - \sum_k [(T_{k1} + \sum_{j=2}^{27} 2T_k j)a_k],$$

$$s.t. \begin{cases} \sum_{j=0}^{30} x_{27j} = 1, \\ \sum_{j=0}^{30} x_{1j} \geq 1, \\ x_{ij} x_{i'j+1} \leq d_{ii'}, \\ V_j \leq 2 + Mx_{12j}, \\ Q_{kj} + T_{kj} - S_{kj} \geq 0, \\ Q_{kj+1} = Q_{kj} + T_{kj} - S_{kj}, \\ \quad 3Q_{1j} + 2Q_{2j} \leq 1200. \end{cases} \tag{8}$$

此模型为 0-1 混合整数规划模型。

4.3 利用深度优先搜索算法求解

4.3.1 事件触发点及有效路径

关注到地图上存在有四个具有特殊性质的节点，即起点、终点、村庄以及矿山，并将其命名为事件触发点。这四个点会对玩家的状态以及资源产生积极的影响。因此，本文不讨论玩家在地图上无意义的徘徊，而认为玩家必然希望以最终顺利到达终点并且资金最多为目标向关键点前进。

通过上述分析，接下来只需要在保证约束成立的条件下讨论若干个事件触发点的排列。进而可以将第一关简化如下图的形式：

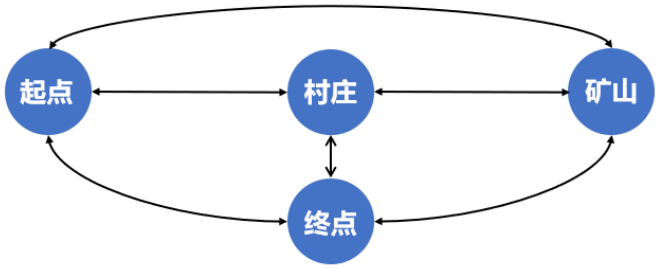


图 2　第一关节点图

此举关注于玩家一最短路径的方式往返于特殊事件点，可以排除众多无效的节点。

4.3.2 深度优先搜索策略

首先构建一个事件树，假设玩家从节点 1 起点 V_1 出发进行搜索，可以选择到达剩下三个特殊事件发生的节点或选择原地停留，重复进行上述搜索策略，并在每次搜索后记录自身状态的变化及玩家剩余水和食物的总量、玩家持有的金钱、距离截止日期的天数。

需要强调，在搜索的过程中，算法优先向下一个状态搜索，直到到达终点或满足剪枝剪枝条件为止，并返回上一个状态，搜索该状态下是否存在其余向下一个阶段迭代的方案。

4.3.3 剪枝策略

在搜索的过程中，往往会寻得不可行解，故需要将其剔除。但若到达终点以后才核实其是否为可行解会使得求解的速度变慢，故本文在每一个搜索的阶段都进行可行性的判别，便可以提早剪枝，增加求解速度。

具体剪枝的判别条件根据上述模型的约束构建：

1. 玩家背包内的水和食物不能超过承载上限；

2. 玩家必须在截至日期之前到达终点。

通过以上策略进行不断的迭代搜索后，可以得出某一种决策序列下玩家到达终点后所剩的金钱，进而与目前记录的最优解进行比较，若优于最优解则重新记录，否则剔除。图3为如何判断剪枝的示意图。

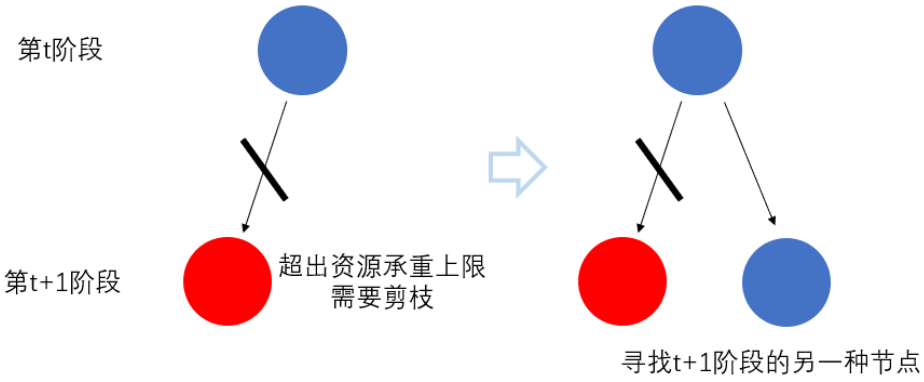
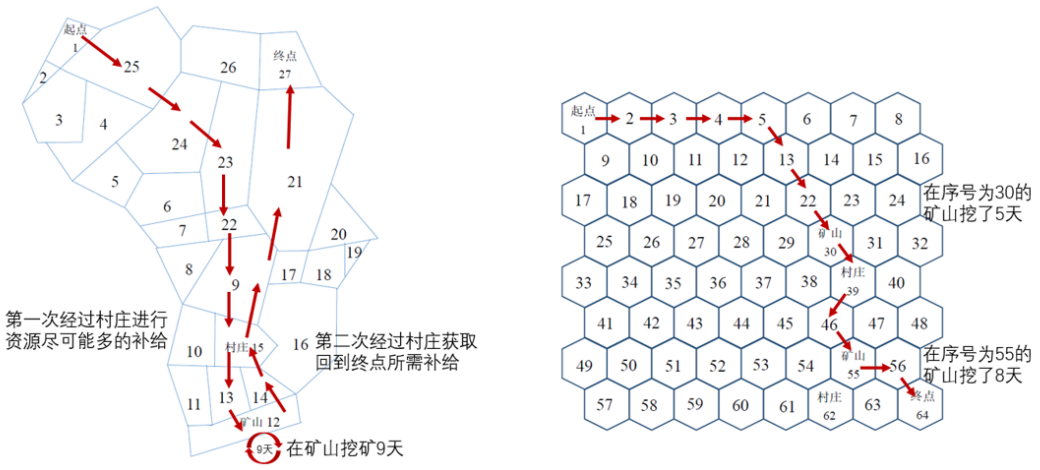


图 3　剪枝策略示意图

4.3.4 求解结果

通过上述算法求解出第一关到达终点时玩家所剩的资金为 10470 元，第二关到达终点时玩家所剩的资金为 12355 元。具体求解结果见附录文件。图4 为第一关和第二关的玩家路径示意图。



(a) 第一关路径　　(b) 第二关路径

图 4　第一关和第二关路径示意图

4.4 结果分析

4.4.1 第一关结果分析

玩家在 24 天便回到了终点，剩余了 6 天空闲时间未能充分利用。但可以发现当前路线下每次补给背包都接近负载上限，可以认为若需要继续挖矿，则需要去村庄额外进行一次资源补给。往返村庄并补给一次资源的时间是 5 天，所以最多还能进行 1 天挖矿。通过计算发现，往返路途中消耗资源对应的资金远大于挖矿一天带来的收益，因此可以基本认为提前回到终点是合理的选择。

4.4.2 第二关结果分析

玩家在第二关中也提前 1 天回到了终点，这与其活动节奏具有一定的关系。观察其路线可以发现，在最后一次补给资源后，玩家并没有仅仅补充返回终点需要的资源，而是将背包装满，并且在途径矿山时尽可能的挖矿。

5 问题二模型的建立与求解

5.1 问题分析

问题二与问题一相比，天气因素不可控。在玩家进行决策时无法预知下一日的天气情况，因此需要根据每一阶段不同的状态以及目标活动进行单阶段决策。关于目标活动的决策，本文拟通过构造以距离和状态作为自变量的回报函数，促进玩家在决策使向目标活动节点靠近。同时，目标活动的选择也需要根据状态进行分析，如：当水和食物资源不足时，应先前往村庄进行补给。另外，关注到转移概率和报酬仅取决于当前状态以及所做出的行为，所以可以被视为是一个马氏过程 [2]。

此外，建立了单阶段决策模型之后，还需要通过蒙特卡洛对整个活动时间段内的天气序列进行模拟。并且让玩家使用上述讨论得到的单阶段决策方法进行全过程仿真，进一步衡量策略的合理性。

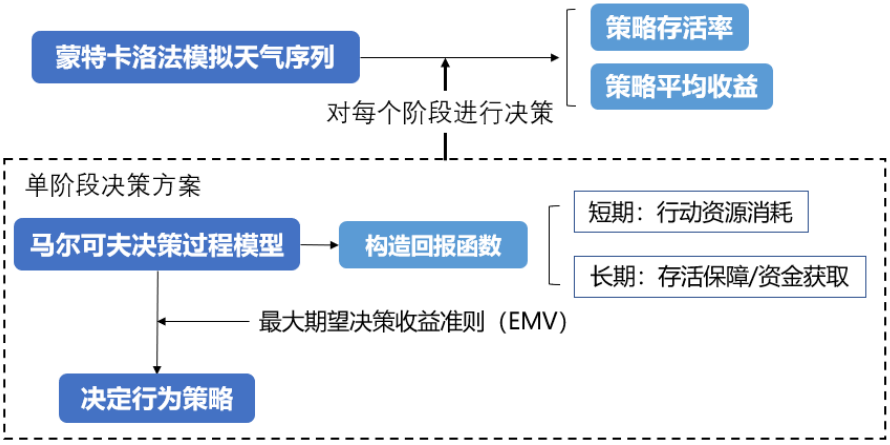


图 5　问题二思路图

5.2 建立风险型决策模型

5.2.1 马尔可夫决策过程

马尔可夫决策过程由一个五元组构成 $M=(T,S,A,P,R)$。

1）决策周期 T： 决策周期又被称为阶段，在本问题中，游戏的每一天为一个决策周期，故有 $T=\{1,2,....,n\}$。其中，n 为游戏的截至日期。

2）状态集 S：

对于此游戏中玩家所处的任意一个状态，可以用状态向量进行表示：

$$\vec{s_t}=[i_t,a_t,b_t,e_t,w_t],t=1,2,...,n,$$

其中，i_t 表示玩家在第 t 天所处的节点，a_t、b_t 以及 e_t 分别表示第 t 天玩家所拥有的水、食物和资金。w_t 表示第 t 天的天气。

对于截至时间为第 n 天的关卡，其状态集 S 中有 n 个状态向量。

3）动作集 A：

考虑到后续中玩家具体移动的方向取决于动作值函数，所以在动作集中不具体讨论移动的方向，仅以“移动”进行表示，同时动作集直接影响了水和食品消耗的倍数，故以具体的数值构成动作集：

$$A=\{1,2,3\}=\cup_{i=1}\alpha_i,$$

其中三个数字分别对应停留原地、移动以及挖矿对应的资源消耗倍数。

对于非矿山，动作集为 $\alpha_i=\{1,2\}$，而对于矿山处的动作集为 $\alpha_{i'}=\{1,2,3\}$。

4）状态转移概率 P：

除天气以外，状态转移仅取决于玩家所处的状态 s_t 以及所作出的动作 a_t，不存在不确定的过程。另外，天气的状态转移概率矩阵不随时间变化，且可以通过关卡一和关卡二的转移频数对相应的概率进行估计。

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}. \tag{9}$$

式中 p_{ij} 为天气类型 i 变换为天气类型 j 的概率。

5）回报函数 R：

本问题中存在两种形式的回报函数，其中一种为正常活动时资源消耗对应的费用，在此称为立即汇报函数；另一种为位于事件触发点处发生资金的改变，包括村庄处交易发生资金的减少、资源的增加，以及在矿山处获得的收益，可以通过效用进行表示。

立即回报函数

立即回报函数取决于玩家所处状态中的天气要素以及行为策略，由于表现为资源的消耗，故取负数，记为 $r(w_t, \alpha_t)$。

表 1　不同天气不同状态下的回报

	晴朗	高温	沙暴
原地逗留	$(-3,-4)$	$(-9,-9)$	$(-10,-10)$
转移	$(-6,-8)$	$(-18,-18)$	\
挖矿	$(-9,-12)$	$(-27,-27)$	$(-30,-30)$

效用函数

与立即回报函数相对，效用函数关注于长期的效果，且效用函数在空间上进行传递，每传递一个单位便会通过折扣因子 τ 削弱。

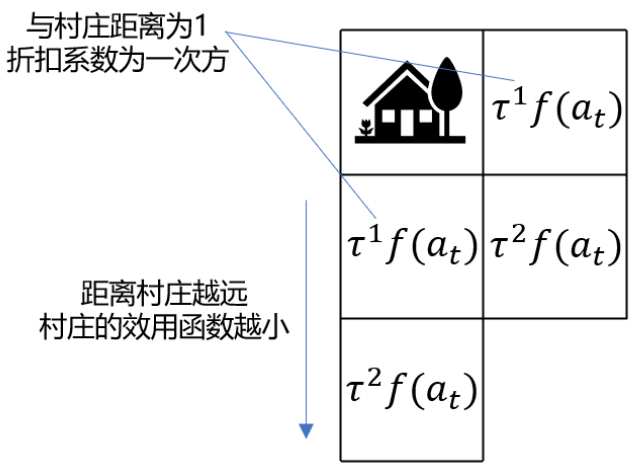


图 6　水 -村庄效用函数传递示意图

折扣因子 τ 也表示决策者对于长远效益的考虑。τ 越大，表示越考虑长远收益。同时，由于效用函数随着距离递减，可以通过设计相应的函数使玩家以获得资源或保证生存为导向进行决策。

根据上文中所提到的三种事件触发点，我们重点讨论其效用函数在形式上的构造。

对于**终点**，其效用与剩余的时间有关，故记为 $f_T(t)$，又可知剩余时间越短，终点效用越大，且当剩余时间接近当前状态到达终点的最短时间，终点必然成为首要前往的事件触发点。因此本文选用了 Sigmoid 函数，当时间处于上述极限状态时，终点的效用值达到一个极大值；而在平时终点的效用值并不明显：

$$f_T(t) = M(1 - \frac{1}{1 + e^{t-(1+\beta)t'(\vec{s})+1}}), \tag{10}$$

其中 $t'(\vec{s})$ 表示当前状态下到达终点的最短时间，增加 1 是为了提前一天对玩家进行传递信息，$(1+\beta)$ 表示由于沙暴导致到达终点时间的延长，β 为沙暴出现的概率。

对于**村庄**，其两个效用函数分别对应水和食物两个资源，基本思想与终点相同，即资源紧缺时，村庄的效用函数值变大。形式与终点基本相同：

$$\begin{aligned} f_A(a_t) &= \gamma_1 M(1 - \frac{1}{1 + e^{a_t-(1+\beta)a'(\vec{s})+1}}), \\ f_B(b_t) &= \gamma_1 M(1 - \frac{1}{1 + e^{b_t-(1+\beta)b'(\vec{s})+1}}), \end{aligned} \tag{11}$$

各变量的含义对应终点效用函数的变量含义。

对于**矿山**，其作为游戏里的资源获取地区，优先级应该低于保障生存的终点和村庄。矿山效益与剩余的时间以及剩余的水和食物有关，当上述资源越多，矿山的效用

值便越高：

$$f_k(a_t,b_t,t)=\gamma_3 l(1-\frac{1}{1+e^{z-(1+\beta)(\min_i z_i+1)}})-3r(w_t,a_t),$$

$$\begin{cases} z_1=a_t-a'(\vec{s})+1, \\ z_2=b_t-b'(\vec{s})+1, \\ z_3=t-t'(\vec{s})+1. \end{cases} \tag{12}$$

其中 $min_i z_i$ 是为了保障生存，考虑最少的资源决定矿山的效用值，l 表示挖矿的基本收益，而 $3r(w_t,a_t)$ 则表示挖矿的基本开销其余变量含义与上文相同。

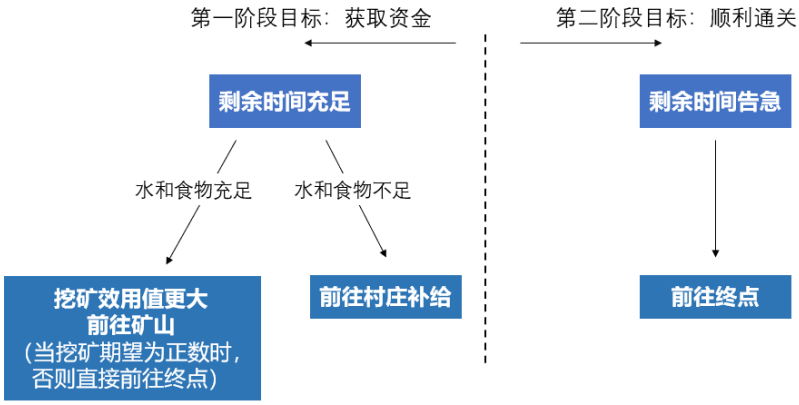


图 7　决策总体思路图

上图为决策模型的总体思路，阶段的划分是根据玩家剩余的时间，实质上也是根据玩家自身的状态进行分析。进而使得每一步决策都是根据自身状态的实时决策。

5.2.2 基于最大期望收益决策准则确定行为策略

本问题中只能知道当天的天气，所以在每一个阶段都需要对采取的行为策略进行决策，本文以最大期望收益作为决策准则构建相应的模型。

事件：本模型中除天气外的所有状态都与策略有关，故以天气作为事件，通过关卡一和关卡二的频数估计相应的频率。

策略：行动集中各元素对应的是状态下不同的策略，不同状态的策略不同，即需要考虑是否在矿山才能决定能否进行挖矿行为。

收益：在上文中，已经给出了各行为的效用函数，结合立即回报函数和效用函数给出每一个动作的效用值。

由此可形成每一步“策略 -事件”的表格：

表 2　策略 -事件

S		天气			EMV
		晴朗	高温	沙暴	
		发生概率			
		p_1	p_2	p_3	
策略	逗留	a_{11}	a_{12}	a_{13}	$\sum_{i=1}^{3} a_{1i}p_j$
	行走	a_{21}	a_{22}	\	$\sum_{i=1}^{3} a_{2i}p_j$
	挖矿	a_{31}	a_{32}	a_{33}	$\sum_{i=1}^{3} a_{3i}p_j$

对应策略为：

$$max_i \sum_j p_j a_{ij} \rightarrow S_k^*.$$

5.3 利用蒙特卡洛法对模型仿真求解

5.3.1 利用蒙特卡洛法对天气进行模拟

问题二的背景中没有给出每日的天气情况，于是本文通过对问题一天气情况的频数对各种天气出现的频率进行估计，之后通过蒙特卡洛法生成若干组天气序列，作为后续模拟的天气条件。

5.3.2 效用值函数计算

1) 建立最短距离矩阵

通过 Floyd 算法求解出每两个区域之间的最短距离，本文重点考虑特殊事件触发点之间的距离。

2) 各状态回报函数的计算

计算玩家在任意区域时的行动总回报函数 (w_t, α_t)：

$$F(\vec{s_k}) = f_T + f_A + f_B + f_k - r(w_t, \alpha_t), \tag{13}$$

此效用函数分别考虑了在该区域时的立即回报函数和矿山、村庄的回报函数。

3) 行动决策

当玩家在某一区域时，可以通过计算总回报效用函数，得出该名玩家每一种行动的效用, 并采取期望最大的一个行为作为他的行动策略。

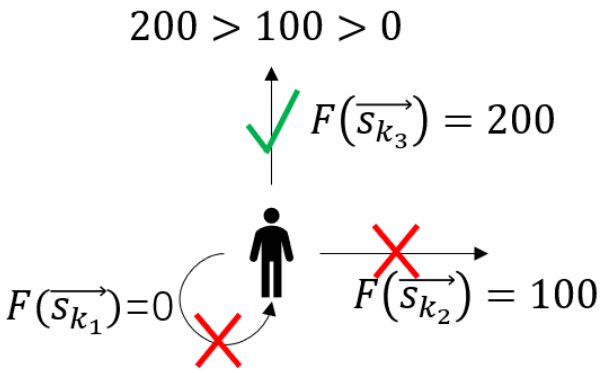


图 8　行为策略选择示意图

上图为玩家行动策略选择示意图，依据的值与状态相关，使玩家能够实时决策。

5.3.3 求解结果

1) 第三关结果：通过蒙特卡洛模拟出一万条不同的天气序列，对每一个天气序列都求一个玩家的决策线路，最后得到玩家的通关率为 99.98%，平均剩余资金为 9377.4 元。

2) 第四关结果：通过模拟出一万条不同天气序列下玩家的顺利到达终点的占比为 88.675%，玩家在到达终点后所剩的金钱数量分布如下图所示：

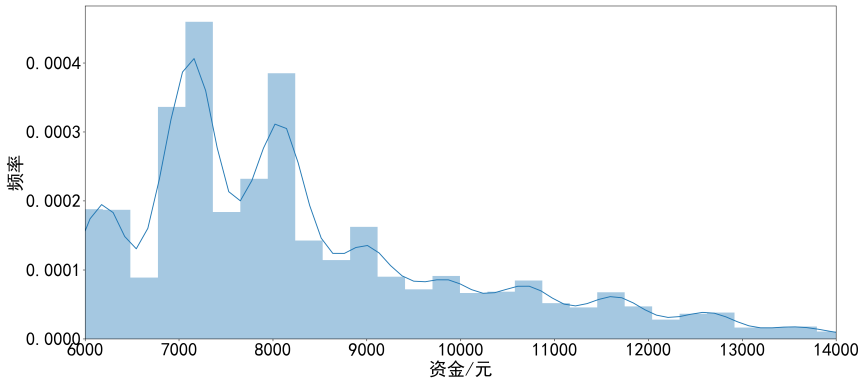


图 9　第四关玩家剩余资金分布图

其中通关玩家的资金分布期望值为 8322.5 元，且通关玩家中所剩资金大于初始值的人数占比为通关人数的 17.99%。通关玩家中所剩资金最大为 18475 元。

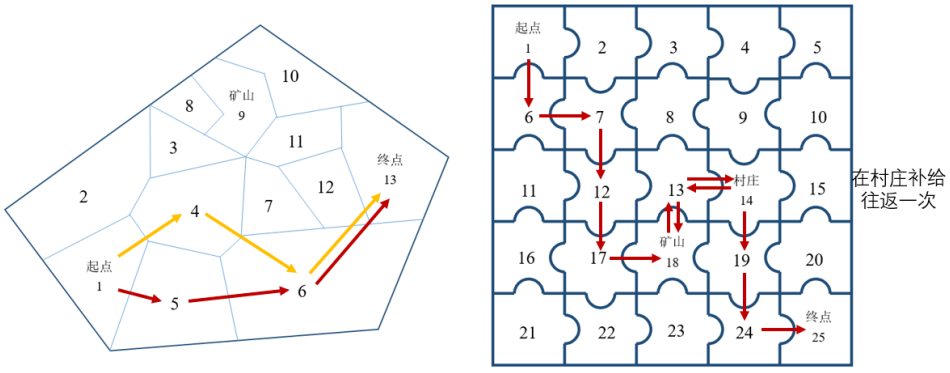
5.4 结果分析

5.4.1 第三关结果分析

观察到模拟结果的路径较为单一，基本为直接前往终点。经过核算发现，在第三关中挖矿的期望收益为负值，故决策的路径符合剩余资金最大化的目标。

5.4.2 第四关结果分析

观察到玩家的行为基本可以总结为以最短路径前往某一事件触发点，因此可以间接说明问题一中对于事件触发点及有效路径假设的合理性。



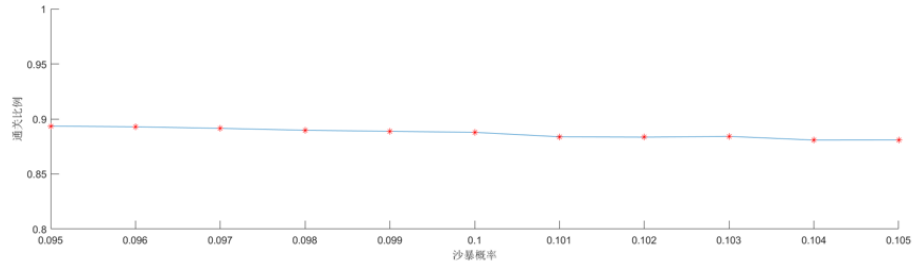
（a）第三关部分路径　　（b）第四关部分路径

图 10　问题二求解示意图

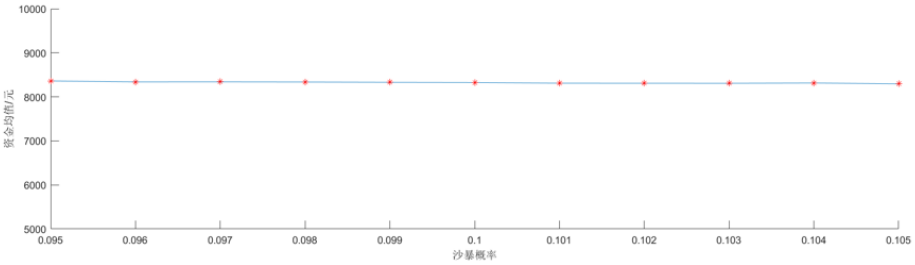
图10中（a）为第三关的路径示意图，图中表示第三关只有两个最优路径。（b）为第四关的路径示意图其中红色路径为其中模拟出的一条路径。

5.5 灵敏度分析

随着第四关沙暴天气出现的概率，玩家行走的路径规划情况会发生变化。于是对第四关模型的沙暴天气出现概率进行灵敏度分析，沙暴的概率做 ±5% 的变化，并比较产生此变化以后对通过比例和剩余金钱均值的影响。得到的影响结果如下图11所示：



(a) 沙暴概率-通关比例



(b) 沙暴概率-平均剩余资金

图 11　灵敏度分析结果图

从灵敏度分析的结果来看，当沙暴发生概率发生 ±5% 的变化时，玩家完成率在 (-0.63% ,0.88%) 的范围内波动，玩家的剩余资金均值则在 (-0.28%,0.36%) 的范围内波动。说明模型具有较强的稳定性，策略较为保守，沙暴发生概率的小范围变化对模型影响较小 。

6 问题三模型的建立与求解

6.1 问题分析

问题三需要额外考虑多名玩家相互博弈的情况。对于第五关，题目要求进行两人单阶段非合作博弈，且决策的内容是接下来完整的路线。注意到若不对问题进行简化，策略集包含有巨量的策略，因此本文结合问题一中事件触发点的思想，仅讨论通过起点、村庄、矿山和终点串联的路径。对于第六关，则为多人多阶段非合作博弈，对于每一步都需要结合当前的状态进行一次博弈，关键在于求解此博弈模型局中人的最佳混合策略。并且根据纳什定理，第六关多人多阶段非合作博弈必然存在平衡局势。

6.2 建立第五关的双人非合作博弈模型

结论：在第五关中无需考虑玩家去矿山挖矿的情况。

假设第 i 天的花费的水和食物总量为 S_i, 且 $S_i = V_i B_i$ 其中 V_i 为第 i 天的行为系数，当玩家原地停留 $V_i = 1$，当玩家前往下一个区域 $V_i = 2$，当玩家在挖矿 $V_i = 3$；B_i 则为基于当天天气的基础消耗量。通过问题一的模型算出玩家前往矿山进行挖矿时 n 天

的最大收益 $M = \sum_{i=1}^{n} S_i$, 结合人工核算的两名玩家同步直接前往终点情况下 n 天的收益 $N = \sum_{i=1}^{n} S_i$，可以得出在任何情况下 $M \leq N$。

因此基于两人博弈的情况下，不需要考虑是否前往矿山，只需考虑尽可能用最少的花费回到终点，于是利用问题一采用的深度优先搜索算法求出在单人情况下的最佳收益路线，最后求得两条行为路线分别为路线 A：$1 \to 4 \to 4 \to 6 \to 13$ 和路线 B：$1 \to 5 \to 5 \to 6 \to 13$。

1) 局中人集合：参与行动的一共有两个玩家，则 $I = 1, 2$。

2) 策略集：在一局对策中，可供玩家选择的一个行动方案称为一个策略。参与策略的每一名玩家都有一个策略集 S_i, 其中策略集均为有限集。在第五关里两名玩家的策略集相同为（路线 1，路线 2，..., 路线 n）。

3) 局势：在一局游戏中，各个玩家选定的策略形成的策略组可成为局势，全体局势的集合 S 可用各个玩家策略集的笛卡尔积表示，即 $S = S_1 \times S_2$

4) 支付函数：分别设玩家 1 和玩家 2 的策略集为 $S_1 = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ 和 $S_2 = \{\beta_1, \beta_2, ..., \beta_n\}$, 则玩家 1 的任意决策 α_i 和玩家 2 的任意决策 α_j 形成局势 s_{ij}。任意一个局势 s_{ij} 最后可以玩家 i 得到一个支付值 $H_i(s)$。

6.3 第五关模型求解

通过求解混合函数的纳什均衡求解玩家 1 和玩家 2 分别选择路线 A 和路线 B 的概率。

玩家 i 的策略空间为 $S_i = \{s_{i1}, s_{i2}\}$, 则玩家 i 以概率分布 $p_j = (p_{i1}, p_{i2})$ 随机在策略空间中选择策略，将其称之为混合策略其中 $0 \leq p_{ij} \leq 1$, 且 $p_{i1} + p_{i2} = 1$

		玩家 1	
		S_{11}	S_{12}
玩家 2	S_{21}	-930,-930	-575,-575
	S_{22}	-575,-575	-930,-930

本次玩家 1 和玩家 2 博弈基于两个原则，1、不能让对方玩家真的自己选择的策略，2、玩家 1 选择路线 A 和路线 B 的概率一定要让玩家 2 的选择路线 A 和路线 B 的期望相等，反之同理。最后可求得玩家 1 和玩家 2 的选择概率矩阵。

玩家 1 的混合策略：

$$p_{11} \times 930 + p_{12} \times 575 = p_{11} \times 575 + p_{12} \times 930$$

玩家 2 的混合策略：

$$p_{21} \times 930 + p_{22} \times 575 = p_{21} \times 575 + p_{22} \times 930$$

且玩家 1 和 2 的选择概率之和为 1。可以算出玩家 1 和玩家 2 分别对路线 A，B 的选择概率如表3所示：

表 3　选择策略概率及期望支付

	策略	期望支付
玩家 1	（0.5，0.5）	752.5
玩家 2	（0.5，0.5）	752.5

表中可知，两名玩家选择线路 A 和 B 的概率分别为 50%, 最后的期望支付为 752.5。

6.4 建立第六关的多人多阶段非合作博弈模型

1) 局中人：本关卡有三名玩家，记为 $U = \{1, 2, 3\}$。

2) 策略集：第 i 个局中人的策略集记为 S_i。每个局中人的策略集都有三个行为，$S_i = \{1, 2, 3\}$ 分别对应原地逗留、行走和挖矿。需要强调，对于任意一个玩家在某个节点上，其可以选择的下一个阶段的节点是一个集合，选择的策略在第二问中已经通过马尔可夫决策过程和最大期望决策准则对下一个阶段选择的节点进行了确定，在此仅以行走作为具体的策略。

3) 局势：$s = (s_1, s_2, ..., s_n) \in S_1 \times S_2 \times \cdots \times S_n$

4) 支付函数：在多阶段的博弈模型中，由于每个阶段的策略都会影响下一个阶段的状态，所以局中人的支付函数不仅仅与所选择的策略有关，还需要考虑局中人在该阶段的状态。局中人的状态可以用第二题状态向量进行表示：

$$\vec{s_{ut}} = [i_{ut}, a_{ut}, b_{ut}, e_{ut}, w_{ut}] u \in U, t = 1, 2, ..., n. \tag{14}$$

虽然基本的支付函数与第二问决策模型中的收益计算方法相同，即选择了策略 s_i 后，基于马尔可夫决策过程和最大期望准则给出了计算在资源和资金上的获得以及开销的情况。但是第二问中并没有考虑到多人具有相同选择、共同挖矿时的情况，在此对其进行修正：

多人同时从节点 p 移动到节点 q

结合状态向量，若存在时间段 t 到 $t+1$，$i_{u,t}$ 满足以下条件，便可认为存在多人同时从节点 p 移动到节点 q：

$$\begin{cases} i_{u,t} = i_{u',t}, \\ i_{u,t+1} = i_{u',t+1}, \\ i_{u,t} \neq i_{u,t+1}, \\ \alpha_{u,t} = \alpha_{u',t} = 2. \end{cases} \tag{15}$$

若满足上述条件，则可以判定表达水和食物消耗的立即回报函数需要修正为 $2n \times r(w_t, \alpha_{i,t})$，其中 n 通过同时移动的人数确定。

多人同时在节点 p 处挖矿

多人同时挖矿的判定为需要满足如下所有条件：

$$\begin{cases} i_{u,t} = i_{u',t}, \\ \alpha_{u,t} = \alpha_{u',t} = 3, \end{cases} \tag{16}$$

若满足上述条件，挖矿的状态转移将被修正为 $\frac{1}{k}\Delta c$。

多人同时在节点 p 处购买物资

多人同时购买物资的判定条件如下，针对水的购买：

$$\begin{cases} i_{u,t} = i_{u',t}, \\ \Delta a_{ut} > 0, \Delta a_{u't} > 0, \end{cases} \tag{17}$$

当上述条件满足时，资金的变化应该被修正为 $\Delta a_{ut} = 4\Delta a_{ut} e_a$，其中 e_a 为一箱水的基本价格。

同理，针对食物的购买：

$$\begin{cases} i_{u,t} = i_{u',t}, \\ \Delta b_{ut} > 0, \Delta b_{u't} > 0, \end{cases} \tag{18}$$

当上述条件满足时，资金的变化应该被修正为 $\Delta a_{ut} = 4\Delta b_{ut} e_b$，其中 e_b 为一箱食物的基本价格。

至此，可以得到多阶段多人非合作博弈模型。

6.5 第六关模型求解

6.5.1 利用蒙特卡洛法对天气进行模拟

与上一问的思路相同，根据关卡一和关卡二的天气频数确定各天气的概率，每个玩家都不知道全部模拟的天气序列，决策仅通过当天天气以及其余状态进行。由于关卡六要求沙暴天气较少，故本文将晴朗、高温以及沙暴的概率分别设定为 30%、60% 和 10%。

6.5.2 基于帝国竞争算法求解博弈模型

给定了天气序列后，通过三名玩家所处的状态可以计算出各局势下每名玩家的收益函数。接下来便需要寻求平衡局势。

根据纳什定理，非合作多人对策在混合策略意义下的平衡局势一定存在 [3]，故依照平衡局势的定义进行搜索。平衡局势的定义如下：

$$E_i(x||z^i) \leq E_i(x), \forall i \in I, z^i \in S_i^*, \tag{19}$$

该式表示任意一个混合局势的变动都不会使得原局势变的更优。进而将其转化为求解函数最小值的问题 [4]：

$$\min v(S) = \sum_{u \in U} \sum_{\alpha \in A} [\frac{|u_i(s_{ij}, p_{-i}) - u_i(p)| + u_i(s_{ij}, p_{-i}) - u_i(p)}{2}]^2, \tag{20}$$

上式中，$u_i(s_{ij}, p_{-i})$ 表示玩家 u_i 采用了纯策略 s_{ij}。

帝国竞争算法 (imperialist competitive algorithm,ICA) 是一种模拟社会政治行为的智能算法，由 Atashpaz-Gargari 等人于 2007 年提出, 可以用于解决优化问题。算法的伪代码如下：

Algorithm 1: Function ImperialistCompetitveAlgorithm(x)

Initialize /* 初始化：设置多个初始国家解 $X_n, n = 1, ..., n_max$。从中选取 N 个最好质量的解使其作为 N 个帝国的殖民国家，$X_1, X_2, ..., X_N$，为每个殖民国家分配殖民国家势力比例数量的殖民地得到 N 个帝国 $E_1, E_2, ..., E_N$ */

iteration = 1;

Repeat for $i \in [1, N]$ **do**

  EmpireAssimilate(E_i) /* 帝国同化 */

  ColoniesRevolve(E_i) /* 殖民地革命 */

  EmpirePosses(E_i) /* 殖民国家替换 */

  CaculateNewCost(E_i) /* 重新计算国家权力 */

end

EmpireUnite($E_1, E_2, \ldots, E_N$) /* 合并相似帝国 */

EmpireCompetition($E_1, E_2, ..., E_N$) /* 帝国竞争，分配殖民地 */

iteration ← iteration+1; 进入下一次迭代

until 只有一个帝国剩下/达到迭代次数。

6.6 第六关策略分析

对于本问题，行为策略的选择取决于根据自身状态计算的效用函数以及对手行为的策略。

6.6.1 效用函数

首先分析效用期望值对于玩家的意义。与第二问相同，效用函数会随着状态的变化而变化。根据题目的要求，顺利通过作为第一要义，故状态中水、食物以及剩余时间便成为决策的首要考虑因素，显然当资源临近缺少时，玩家会根据效用函数空间传递的特性主动向村庄或终点靠近。但生存资源还未告急时，村庄和终点并不具有相应的吸引力，故作为第一要义的生存没有产生急切的需求，因此考虑作为第二要义的资金获取。

同样根据效用函数空间传递的特性，即随着距离依照折扣系数递减，若推算在矿山挖矿的期望收益为正值，玩家便会主动向矿山靠近，否则仍旧向终点前进。

此外，本问还对多人同步动作的情况进行了修正，在一定程度上可以认为是对相应行为的惩罚。

6.6.2 博弈策略

由于第五关和第六关的玩家都处于竞争状态，因此每个玩家采取博弈策略时都只能认为自己处于最不利状态，故只能将目标确定为使自己的收益不少于某个值。对于本问题所处于的静态博弈，关注稳定状态，即为做出的策略使得任何一个玩家自行改变其混合策略都会使得自身效益期望值小于或等于原期望值。

考虑到所有玩家都是理性的假设，我们有理由希望能够到达这种稳定状态。

7 模型的总结与评价

7.1 模型优点

1. 采用了帝国竞争算法求解多人博弈模型，算法收敛速度快，收敛精度高。

2. 通过蒙特卡洛模拟了玩家在游戏中的实际行动，并通过通关率和平均剩余资金验证了方案的有效性。

3. 建立了马尔可夫决策过程模型，使得玩家在未知天气的情况下能够实时根据状态判断接下来是否需要前往村庄补给或是前往终点，能够平衡目标地点的长远效益以及行动策略带来的即时花费。

7.2 模型缺点

1. 第六关的模型较为复杂，使游戏模拟程序运行时间较长。

7.3 模型改进

对于未知天气的情况，若能够获得精确的天气概率，蒙特卡洛的结果将会更为精准。此外，本文在第五关的博弈过程中仅采纳了几条较为典型的路径作为策略集，若时间允许，可以讨论更多的策略，使模型更全面。

参考文献

[1] 庞博. 动态网络中的流问题 [D]. 国防科学技术大学,2009.

[2] 刘克. 实用马尔可夫决策过程 [M]. 清华大学出版社, 2004.

[3] Shah S M , Borkar V S . Q-learning for Markov decision processes with a satisfiability criterion[J]. Systems & Control Letters, 2018, 113:45-51.

[4] Ott, Jonathan Theodor. A Markov decision model for a surveillance application and risk-sensitive Markov decision processes[J]. 2010.

[5] Brunette M , Laye J A . Optimizing forest management under storm risk with Markov decision process model[J]. Post-Print, 2014, 4(2):141-163.

[6] 贾文生, 向淑文, 杨剑锋, 等. 基于免疫粒子群算法的非合作博弈 Nash 均衡问题求解 [J]. 计算机应用研究, 2012(01):28-31.

[7] Pavlidis N G , Parsopoulos K E , Vrahatis M N . Computing Nash equilibria through computational intelligence methods[J]. Journal of Computational and Applied Mathematics, 2005.

[8] Francisco, FacchineiChristian, Kanzow. Generalized Nash equilibrium problems[J]. 4or, 2007.

[9] 陆栋梁. 多人博弈与合作演化研究 [D]. 苏州大学,2012.

附录 A

附件清单

Result.xlsx: 关卡一和关卡二的结果
p1main.m: 关卡一仿真主程序
dfs: 关卡一深度优先搜索函数
p2main.m: 关卡二仿真主程序
dfs2.m: 关卡二深度优先搜索函数
p4main.m: 关卡三四仿真主程序
compute,value.m: 关卡三四计算效用值函数
p4sensitymain.m: 关卡三四灵敏度分析主程序
p4sensity.m: 关卡三四灵敏度分析子函数
ICA：帝国竞争算法求解关卡六

1.1 问题一结果

第一关

日期	所在区域	剩余资金数	剩余水量	剩余食物量
0	1	5780	178	333
1	25	5780	162	321
2	24	5780	146	309
3	23	5780	136	295
4	23	5780	126	285
5	22	5780	116	271
6	9	5780	100	259
7	9	5780	90	249
8	15	4150	243	235
9	14	4150	227	223
10	12	4150	211	211
11	12	5150	181	181
12	12	6150	157	163
13	12	7150	142	142
14	12	7150	118	124
15	12	9150	94	106
16	12	10150	70	88
17	12	10150	60	78
18	12	10150	50	68
19	12	11150	26	50
20	14	11150	10	38
21	15	10470	36	40
22	9	10470	26	26
23	21	10470	10	14
24	27	10470	0	0
25				
26				
27				
28				
29				
30				

第二关

日期	所在区域	剩余资金数	剩余水量	剩余食物量
0	1	6475	247	229
1	2	6475	231	217
2	3	6475	215	205
3	4	6475	205	191
4	4	6475	195	181
5	5	6475	185	167
6	13	6475	169	155
7	13	6475	159	145
8	22	6475	149	131
9	30	6475	133	119
10	30	7475	109	101
11	30	8475	79	71
12	30	9475	55	53
13	30	10475	40	32
14	30	11475	16	14
15	39	4355	240	238
16	47	4355	224	226
17	47	4355	214	216
18	47	4355	204	206
19	55	4355	188	194
20	55	5355	164	176
21	55	6355	149	155
22	55	7355	134	134
23	55	8355	110	116
24	55	9355	95	95
25	55	10355	65	65
26	55	11355	41	47
27	55	12355	26	26
28	56	12355	16	12
29	64	12355	0	0
30				

1.2 matlab 源程序

1.2.1 p1main

```
clear, clc;
global final_results;
global max_money;
%global check;
max_money = 0;
%check = 0;
m = zeros(27, 27);
m(1,2) = 1;
m(1,25) = 1;
m(2,1) = 1;
m(2,3) = 1;
m(3,2) = 1;
m(3,4) = 1;
m(3,25) = 1;
m(4,[3,5,24,25]) = 1;
m(5,[4,6,24]) = 1;
m(6,[5,7,23,24]) = 1;
m(7,[6,8,22]) = 1;
m(8,[7,9,22]) = 1;
m(9,[8,10,15,16,17,21,22]) = 1;
m(10, [9,11,13,15]) = 1;
m(11, [10,12,13]) = 1;
m(12, [11, 13, 14]) = 1;
m(13, [10,11,12,14,15]) = 1;
m(14, [12,13,15,16]) = 1;
m(15, [9, 10, 13, 14, 16]) = 1;
m(16, [9, 14, 15, 17, 18]) = 1;
m(17, [9, 16, 18, 21]) = 1;
m(18, [16, 17, 19, 20]) = 1;
m(19, [18, 20]) = 1;
m(20, [18, 19, 21]) = 1;
m(21, [20, 17, 9, 22, 23, 27]) = 1;
m(22, [23, 7, 8, 9, 21]) = 1;
m(23, [26, 21, 22, 6, 24]) = 1;
```

```
m(24, [4, 5, 6, 23, 25, 26]) = 1;
m(25, [1,3,4,24,26]) = 1;
m(26, [25,24,23,27]) = 1;
m(27, [26, 21]) = 1;

route{1}=[25,24,23,22,9,15]; % 起点到村庄
route{2}=[14,12]; % 村庄到矿山
route{3}=[14,15]; % 矿山到村庄
route{4}=[25,26,27]; % 起点到终点
route{5}=[25,24,23,9,15,14,12]; % 起点到矿山
route{6}=[9,21,27]; % 村庄到终点
route{7}=[14,15,9,21,27]; % 矿山到终点

result(1, 1) = 1;
result(1, 2) = 10000;
result(1, 3) = 0;
result(1, 4) = 0;
final_results = result;
dfs(0, 1, result, 0);
```

1.2.2 p2main

```
clear, clc;
global final_results;
global max_money;
%global check;
max_money = 0;
%check = 0;
result(1, 1) = 1;
result(1, 2) = 10000;
result(1, 3) = 0;
result(1, 4) = 0;
final_results = result;
dfs2(0, 1, result, 0);
```

1.2.3 p3main

```
clear, clc;
global m;
m = ones(13, 13);
m = m * 100000;
m(1,[2,4,5]) = 1;
m(2,[1,3,4]) = 1;
m(3,[2,4,8,9]) = 1;
m(4,[1,2,3,7,6,5]) = 1;
m(5, [1,4,6]) = 1;
m(6, [5, 4, 7, 12, 13]) = 1;
m(7, [4, 6, 12, 11]) = 1;
m(8, [3, 9]) = 1;
m(9, [3, 8, 10, 11]) = 1;
m(10, [9, 11, 13]) = 1;
m(11, [7, 12, 13, 10, 9]) = 1;
m(12, [6, 7, 11, 13]) = 1;
m(13, [6, 10, 11, 12]) = 1;
for i = 1:13
    m(i, i) = 0;
end
for k = 1:13
    for i = 1:13
        for j = 1:13
            if m(i,k)+m(k,j) < m(i,j)
                m(i, j) = m(i, k) + m(k, j);
            end
        end
    end
end

%% 开始模拟
water = [3, 5, 4, 9, 10];
food = [2, 10, 4, 9, 10];
% 第天0
result(1, 1) = 1;
```

```
result(1, 2) = 10000;
result(1, 3) = 0;
result(1, 4) = 0;

cur_loc = 1;
cur_day = 0;
while true
    cur_day = cur_day + 1;
    if rand > 0.5
        cur_weather = 1;
    else
        cur_weather = 2;
    end
    available_places = find(m(cur_loc, :));
    for k = 1:length(available_places)
        values(k) = compute_value(available_places(k));
```

1.3 p4main

```
clear, clc;
global m;
m = ones(25, 25);
m = m * 10000;
m(1, [2,6]) = 1;
m(5, [4,10]) = 1;
m(21, [16, 22]) = 1;
m(25, [20, 24]) = 1;
for i = 2:4
    m(i, [i-1,i+1,i+5]) = 1;
end
for i = 22:24
    m(i, [i-5,i-1,i+1]) = 1;
end
for i = 6:5:16
    m(i, [i-5, i+5, i+1]) = 1;
end
```

```
for i = 10:5:20
    m(i, [i-5,i+5,i-1])=1;
end
xxx = [7,8,9,12,13,14,17,18,19];
for i = 1:length(xxx)
    m(xxx(i), [xxx(i)-5,xxx(i)-1,xxx(i)+1,xxx(i)+5]) = 1;
end

for i = 1:25
    m(i, i) = 0;
end
for k = 1:25
    for i = 1:25
        for j = 1:25
            if m(i,k)+m(k,j) < m(i,j)
                m(i, j) = m(i, k) + m(k, j);
            end
        end
    end
end


failure_cnt1 = 0;
failure_cnt2 = 0;
failure_cnt3 = 0;
failure_cnt4 = 0;
failure_cnt5 = 0;
failure_cnt6 = 0;
failure_cnt7 = 0;
failure_cnt8 = 0;
failure_cnt9 = 0;
success_cnt = 0;
max_money = 0;

%% 开始模拟
for kkk = 1:100000
    flag1 = 0; % 是否是初始点购买物资
```

```
flag = 0;
water = [3, 5, 3, 9, 10];
food = [2, 10, 4, 9, 10];
for i = 1:30
    weather_p = rand;
    if weather_p > 0 && weather_p < 0.3
        weather(i) = 1;
    elseif weather_p >= 0.3 && weather_p < 0.9
        weather(i) = 2;
    else
        weather(i) = 3;
    end
end
% 第天0
result = zeros(1, 4);
result(1, 1) = 1;
result(1, 2) = 10000;
result(1, 3) = 0;
result(1, 4) = 0;

cur_loc = 1;
cur_day = 0;

for i = 1:30
    cur_day = cur_day + 1;
    cur_weather = weather(i);
    % 判断天气
    if cur_weather == 3
        cur_loc = cur_loc;
        if cur_loc == 14 % 如果在村庄, 那么先吃饭, 后补给
            result(cur_day+1, 1) = cur_loc;
            result(cur_day+1, 3) = result(cur_day, 3) +
                water(cur_weather+2);
            result(cur_day+1, 4) = result(cur_day, 4) +
                food(cur_weather+2);
            % TODO判断这个时候还有没有东西吃:
```

```
            if water(1)*result(cur_day+1, 3) +
                food(1)*result(cur_day+1,4) > 1200
                status = 0;
                failure_cnt1 = failure_cnt1 + 1;
                flag = 1;
                break;
            end
            result(cur_day+1, 2) = result(cur_day, 2) -
                2*water(2)*result(cur_day+1, 3) -
                2*food(2)*result(cur_day+1,4);
            result(cur_day+1, 3) = 0;
            result(cur_day+1, 4) = 0;
        elseif cur_loc == 18 % 如果被困在矿山, 不挖矿
            result(cur_day+1, 1) = cur_loc;
            result(cur_day+1, 2) = result(cur_day, 2);
            result(cur_day+1, 3) = result(cur_day, 3) +
                water(cur_weather+2);
            result(cur_day+1, 4) = result(cur_day, 4) +
                food(cur_weather+2);
            % TODO判断这个时候有没有东西吃:
            if water(1)*result(cur_day+1, 3) +
                food(1)*result(cur_day+1,4) > 1200
                status = 0;
                failure_cnt2 = failure_cnt2 + 1;
                flag = 1;
                break;
            end
        else % 在路上被困住了
            result(cur_day+1, 1) = cur_loc;
            result(cur_day+1, 2) = result(cur_day, 2);
            result(cur_day+1, 3) = result(cur_day, 3) +
                water(cur_weather+2);
            result(cur_day+1, 4) = result(cur_day, 4) +
                food(cur_weather+2);
            % TODO判断这个时候有没有东西吃:
            if water(1)*result(cur_day+1, 3) +
                food(1)*result(cur_day+1,4) > 1200
```

```
                status = 0;
                failure_cnt3 = failure_cnt3 + 1;
                flag = 1;
                break;
            end
        end
    else % 不是沙暴天气
        available_places = find(m(cur_loc, :) == 1);
        available_places = [available_places, cur_loc];
        values = [];
        for k = 1:length(available_places)
            % 下面这一行可能有点问题1
            values(k) = compute_value(available_places(k), cur_day,
                result(cur_day, 3), result(cur_day, 4));
        end
        [x, y] = max(values); % 可能有多个y
        y = available_places(y(1)); % 选择第一个, 之后可以换成随机选择一个
        result(cur_day+1, 1) = y; % 移动
        cur_loc = y;
        if result(cur_day+1, 1) == 14 % 如果移动到村庄
            result(cur_day+1, 3) = result(cur_day, 3) +
                2*water(cur_weather+2);
            result(cur_day+1, 4) = result(cur_day, 4) +
                2*food(cur_weather+2);
            if water(1)*result(cur_day+1, 3) +
                food(1)*result(cur_day+1,4) > 1200
                status = 0;
                failure_cnt4 = failure_cnt4 + 1;
                flag = 1;
                break;
            end
            if flag1 == 0
                result(cur_day+1, 2) = result(cur_day, 2) -
                    water(2)*result(cur_day+1, 3) -
                    food(2)*result(cur_day+1,4);
                flag1 = 1;
            else
```

```
            result(cur_day+1, 2) = result(cur_day, 2) -
                2*water(2)*result(cur_day+1, 3) -
                2*food(2)*result(cur_day+1,4);
        end
        %result(cur_day+1, 2) = result(cur_day, 2) -
            2*water(2)*result(cur_day+1, 3) -
            2*food(2)*result(cur_day+1,4);
        result(cur_day+1, 3) = 0;
        result(cur_day+1, 4) = 0;
    elseif result(cur_day+1, 1) == 18 && result(cur_day, 1) ~= 18% 如
        果移动到矿山, 第一天无法挖矿
        result(cur_day+1, 3) = result(cur_day, 3) +
            2*water(cur_weather+2);
        result(cur_day+1, 4) = result(cur_day, 4) +
            2*food(cur_weather+2);
        result(cur_day+1, 2) = result(cur_day, 2);
        if water(1)*result(cur_day+1, 3) +
            food(1)*result(cur_day+1,4) > 1200
            status = 0;
            failure_cnt5 = failure_cnt5 + 1;
            flag = 1;
            break;
        end
    elseif result(cur_day+1, 1) == 18 && result(cur_day, 1) == 18 %
        从矿山到矿山, 那么是挖
        矿
        result(cur_day+1, 2) = result(cur_day, 2) + 1000;
        result(cur_day+1, 3) = result(cur_day, 3) +
            3*water(cur_weather+2);
        result(cur_day+1, 4) = result(cur_day, 4) +
            3*food(cur_weather+2);
        if water(1)*result(cur_day+1, 3) +
            food(1)*result(cur_day+1,4) > 1200
            status = 0;
            failure_cnt6 = failure_cnt6 + 1;
            flag = 1;
            break;
        end
```

```
        elseif result(cur_day+1, 1) == 25 % 如果到达终点
            status = 1;
            result(cur_day+1, 3) = result(cur_day, 3) +
                2*water(cur_weather+2);
            result(cur_day+1, 4) = result(cur_day, 4) +
                2*food(cur_weather+2);
            if water(1)*result(cur_day+1, 3) +
                food(1)*result(cur_day+1,4) > 1200
                status = 0;
                failure_cnt7 = failure_cnt7 + 1;
                flag = 1;
                break;
            end
            result(cur_day+1, 2) = result(cur_day, 2) -
                2*water(2)*result(cur_day+1, 3) -
                2*food(2)*result(cur_day+1, 4);
            result(cur_day+1, 4) = 0;
            result(cur_day+1, 3) = 0;
            if result(cur_day+1, 2) > max_money
                max_money = result(cur_day+1, 2);
                final_result = [];
                final_result = result;
            end
            success_cnt = success_cnt + 1;
            earn(success_cnt) = result(cur_day+1, 2);
            flag = 1;
            break;
        else % 如果不是特殊点
            result(cur_day+1, 3) = result(cur_day, 3) +
                2*water(cur_weather+2);
            result(cur_day+1, 4) = result(cur_day, 4) +
                2*food(cur_weather+2);
            result(cur_day+1, 2) = result(cur_day, 2);
            if water(1)*result(cur_day+1, 3) +
                food(1)*result(cur_day+1,4) > 1200
                status = 0;
                failure_cnt8 = failure_cnt8 + 1;
```

```
                    flag = 1;
                    break;
                end
            end
        end
    end
    if flag == 0
        failure_cnt9 = failure_cnt9 + 1;
        bad_results{failure_cnt9} = result;
    end
end
hist(earn);
mean(earn)
```

1.4 dfs

```
function dfs(cur_day, cur_loc, result, flag)
cur_day;
cur_loc;
global final_results
global max_money
%global check;
% 表示是否是第一次来商店flag
water = [3,5,5,8,10];
food = [2,10,7,6,10];

days = [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];

route{1}=[25,24,23,22,9,15]; % 起点到村庄
route{2}=[14,12]; % 村庄到矿山
route{3}=[14,15]; % 矿山到村庄
%route{4}=[25,26,27]; % 起点到终点
route{5}=[25,24,23,22, 9,15,14,12]; % 起点到矿山
route{6}=[9,21,27]; % 村庄到终点
route{7}=[14,15,9,21,27]; % 矿山到终点
route{8}=[12]; % 矿山到矿山
```

```
route{9}=[12]; % 矿山到矿山，但是我休息
% 为起点，为村庄，为矿山，为终点1151227

if cur_loc == 1
    to = [1,5];
    for i = 1:2
        new_result = result;
        new_day = cur_day;
        temp_route = route{to(i)};
        cnt = 1;
        while cnt <= length(route{to(i)})
            new_day = new_day + 1;
            if new_day > 30 % 超出天30
                status = 0;
                return;
            end
            if days(new_day) ~= 3 % 不是沙暴天气
                cur_loc = temp_route(cnt);
                cnt = cnt + 1;
                new_result(new_day+1, 1) = cur_loc; % 当前的位置
                new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                    前一天相同，矿山第一天不能挣钱
                new_result(new_day+1, 3) = new_result(new_day, 3) +
                    2*water(days(new_day)+2);
                new_result(new_day+1, 4) = new_result(new_day, 4) +
                    2*food(days(new_day)+2);
            else
                cur_loc = cur_loc;
                new_result(new_day+1, 1) = cur_loc;
                new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                    前一天相同，矿山第一天不能挣钱
                new_result(new_day+1, 3) = new_result(new_day, 3) +
                    water(days(new_day)+2);
                new_result(new_day+1, 4) = new_result(new_day, 4) +
                    food(days(new_day)+2);
            end
        end
        % 判断是否失败
```

```
        if new_day > 30
            status = 0;
            continue;
        end
        if new_day == 30 && new_result(end, 1) ~= 27
            status = 0;
            continue;
        end
        if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
            status = 0;
            continue;
        end
        % 触发事件到达商店，清零吃掉的水和食物
        if new_result(end, 1) == 15
            if flag == 0 % 如果是第一次来村庄
                new_result(end, 2) = new_result(end, 2) -
                    water(2)*new_result(end, 3) - food(2)*new_result(end, 4);
                new_result(end, 3) = 0;
                new_result(end, 4) = 0;
                flag = 1;
            else
                new_result(end, 2) = new_result(end, 2) -
                    2*water(2)*new_result(end, 3) - 2*food(2)*new_result(end,
                    4);
                new_result(end, 3) = 0;
                new_result(end, 4) = 0;
            end
        end
        dfs(new_day, cur_loc, new_result, flag);
    end
elseif cur_loc == 15
    to = [2, 6];
    for i = 1:2
        new_result = result;
        new_day = cur_day;
        temp_route = route{to(i)};
        cnt = 1;
```

```
        while cnt <= length(route{to(i)})
            new_day = new_day + 1;
            if new_day > 30 % 超出天30
                status = 0;
                return;
            end
            if days(new_day) ~= 3 % 不是沙暴天气
                cur_loc = temp_route(cnt);
                cnt = cnt + 1;
                new_result(new_day+1, 1) = cur_loc; % 当前的位置
                new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                    前一天相同，矿山第一天不能挣钱
                new_result(new_day+1, 3) = new_result(new_day, 3) +
                    2*water(days(new_day)+2);
                new_result(new_day+1, 4) = new_result(new_day, 4) +
                    2*food(days(new_day)+2);
            else
                cur_loc = cur_loc;
                new_result(new_day+1, 1) = cur_loc;
                new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                    前一天相同，矿山第一天不能挣钱
                new_result(new_day+1, 3) = new_result(new_day, 3) +
                    water(days(new_day)+2);
                new_result(new_day+1, 4) = new_result(new_day, 4) +
                    food(days(new_day)+2);
            end
        end
        % 判断是否失败
        if new_day > 30
            status = 0;
            continue;
        end
        if new_day == 30 && new_result(end, 1) ~= 27
            status = 0;
            continue;
        end
        if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
            status = 0;
```

```
            continue;
        end
        % 触发事件到达商店，清零吃掉的水和食物
        if new_result(end, 1) == 15
            if flag == 0 % 如果是第一次来村庄
                new_result(end, 2) = new_result(end, 2) -
                    water(2)*new_result(end, 3) - food(2)*new_result(end, 4);
                new_result(end, 3) = 0;
                new_result(end, 4) = 0;
                flag = 1;
            else
                new_result(end, 2) = new_result(end, 2) -
                    2*water(2)*new_result(end, 3) - 2*food(2)*new_result(end,
                    4);
                new_result(end, 3) = 0;
                new_result(end, 4) = 0;
            end
        end
        dfs(new_day, cur_loc, new_result, flag);
    end
elseif cur_loc == 12
    to = [3, 7, 8, 9];
    % 在矿山挖矿
    for i = 3:3
        new_result = result;
        new_day = cur_day;
        temp_route = route{to(3)};
        new_day = new_day + 1;
        if new_day > 30 % 超出天30
            status = 0;
            return;
        end
        % 不管是不是沙尘暴，我都要挖矿
        cur_loc = temp_route(1);
        new_result(new_day+1, 1) = cur_loc;
        new_result(new_day+1, 2) = new_result(new_day, 2) + 1000;
```

```
        new_result(new_day+1, 3) = new_result(new_day, 3) +
            3*water(days(new_day)+2);
        new_result(new_day+1, 4) = new_result(new_day, 4) +
            3*food(days(new_day)+2);
        % 判断是否失败
        if new_day > 30
            status = 0;
            continue;
        end
        if new_day == 30 && new_result(end, 1) ~= 27
            status = 0;
            continue;
        end
        if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
            status = 0;
            continue;
        end
        dfs(new_day, cur_loc, new_result, flag);
    end
    % 在矿山不挖矿
    for i = 4:4
        new_result = result;
        new_day = cur_day;
        temp_route = route{to(4)};
        new_day = new_day + 1;
        if new_day > 30 % 超出天30
            status = 0;
            return;
        end
        % 不挖矿，只休息
        cur_loc = temp_route(1);
        new_result(new_day+1, 1) = cur_loc;
        new_result(new_day+1, 2) = new_result(new_day, 2);
        new_result(new_day+1, 3) = new_result(new_day, 3) +
            water(days(new_day)+2);
        new_result(new_day+1, 4) = new_result(new_day, 4) +
            food(days(new_day)+2);
```

```
        % 判断是否失败
        if new_day > 30
            status = 0;
            continue;
        end
        if new_day == 30 && new_result(end, 1) ~= 27
            status = 0;
            continue;
        end
        if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
            status = 0;
            continue;
        end
        dfs(new_day, cur_loc, new_result, flag);
    end
    for i = 1:2
        new_result = result;
        new_day = cur_day;
        temp_route = route{to(i)};
        cnt = 1;
        while cnt <= length(route{to(i)})
            new_day = new_day + 1;
            if new_day > 30 % 超出天30
                status = 0;
                return;
            end
            if days(new_day) ~= 3 % 不是沙暴天气
                cur_loc = temp_route(cnt);
                cnt = cnt + 1;
                new_result(new_day+1, 1) = cur_loc; % 当前的位置
                new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                    前一天相同，矿山第一天不能挣钱
                new_result(new_day+1, 3) = new_result(new_day, 3) +
                    2*water(days(new_day)+2);
                new_result(new_day+1, 4) = new_result(new_day, 4) +
                    2*food(days(new_day)+2);
            else
                cur_loc = cur_loc;
```

```
            new_result(new_day+1, 1) = cur_loc;
            new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                前一天相同，矿山第一天不能挣钱
            new_result(new_day+1, 3) = new_result(new_day, 3) +
                water(days(new_day)+2);
            new_result(new_day+1, 4) = new_result(new_day, 4) +
                food(days(new_day)+2);
        end
    end
    % 判断是否失败
    if new_day > 30
        status = 0;
        continue;
    end
    if new_day == 30 && new_result(end, 1) ~= 27
        status = 0;
        continue;
    end
    if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
        status = 0;
        continue;
    end
    % 触发事件到达商店，清零吃掉的水和食物
    if new_result(end, 1) == 15
        if flag == 0 % 如果是第一次来村庄
            new_result(end, 2) = new_result(end, 2) -
                water(2)*new_result(end, 3) - food(2)*new_result(end, 4);
            new_result(end, 3) = 0;
            new_result(end, 4) = 0;
            flag = 1;
        else
            new_result(end, 2) = new_result(end, 2) -
                2*water(2)*new_result(end, 3) - 2*food(2)*new_result(end,
                4);
            new_result(end, 3) = 0;
            new_result(end, 4) = 0;
        end
    end
```

```
        dfs(new_day, cur_loc, new_result, flag);
    end

else % 到达了终点
    final_money = result(end, 2) - 2*water(2)*result(end, 3) -
        2*food(2)*result(end, 4);
%    if check == 0
%        result
%        check = 1;
%    end
    final_money;
    result(end, 2) = final_money;
    if final_money > max_money
        final_results = result;
        max_money = final_money;
        max_money;
    end
    return;
end
```

1.5 dfs2

```
function dfs2(cur_day, cur_loc, result, flag)
cur_day;
cur_loc;
global final_results
global max_money
%global check;
% 表示是否是第一次来商店flag
water = [3,5,5,8,10];
food = [2,10,7,6,10];

days = [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];

route{1}=[2,3,4,5,13,22,30]; % 起点到矿山1
route{2}=[2,3,4,5,13,22,30,39]; % 起点到村庄1
```

```
route{3}=[2,3,4,12,21,29,38,46,55]; % 起点到矿山2
route{4}=[2,3,4,12,21,29,28,46,55,62]; % 起点到村庄2
route{5}=[39]; % 矿山到村庄11
route{6}=[30]; % 在矿山挖矿1
route{7}=[30]; % 在矿山休息1
route{8}=[39,47,55]; % 矿山到矿山12
route{9}=[39,47,55,62]; % 矿山到村庄12
route{10}=[39,47,56,64]; % 矿山到终点1
route{11}=[30]; %村庄到矿山11
route{12}=[47, 55]; %村庄到矿山12
route{13}=[47, 55, 62]; % 村庄到村庄12
route{14}=[47,56,64]; % 村庄到终点1
route{15}=[47,39,30]; % 矿山到矿山21
route{16}=[47,39]; % 矿山到村庄21
route{17}=[62]; % 矿山到村庄22
route{18}=[56,64]; % 矿山到终点2
route{23}=[55]; % 矿山挖矿2
route{24}=[55]; % 矿山休息2
route{19}=[55]; % 村庄到矿山22
route{20}=[55, 47, 39]; % 村庄到村庄21
route{21}=[55, 47, 39, 30]; % 村庄到矿山21
route{22}=[63, 64]; % 村庄到终点2

%为起点，为村庄，为矿山，为村庄，为矿山，为终点139130162255264

if cur_loc == 1
    to = [1,2,3,4];
    for i = 1:4
        new_result = result;
        new_day = cur_day;
        temp_route = route{to(i)};
        cnt = 1;
        while cnt <= length(route{to(i)})
            new_day = new_day + 1;
            if new_day > 30 % 超出天30
                status = 0;
                return;
```

```
            end
            if days(new_day) ~= 3 % 不是沙暴天气
                cur_loc = temp_route(cnt);
                cnt = cnt + 1;
                new_result(new_day+1, 1) = cur_loc; % 当前的位置
                new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                    前一天相同，矿山第一天不能挣钱
                new_result(new_day+1, 3) = new_result(new_day, 3) +
                    2*water(days(new_day)+2);
                new_result(new_day+1, 4) = new_result(new_day, 4) +
                    2*food(days(new_day)+2);
            else
                cur_loc = cur_loc;
                new_result(new_day+1, 1) = cur_loc;
                new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                    前一天相同，矿山第一天不能挣钱
                new_result(new_day+1, 3) = new_result(new_day, 3) +
                    water(days(new_day)+2);
                new_result(new_day+1, 4) = new_result(new_day, 4) +
                    food(days(new_day)+2);
            end
        end
        % 判断是否失败
        if new_day > 30
            status = 0;
            continue;
        end
        if new_day == 30 && new_result(end, 1) ~= 64
            status = 0;
            continue;
        end
        if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
            status = 0;
            continue;
        end
        % 触发事件到达商店，清零吃掉的水和食物
        if new_result(end, 1) == 39 || new_result(end, 1) == 62
            if flag == 0 % 如果是第一次来村庄
```

```
                new_result(end, 2) = new_result(end, 2) -
                    water(2)*new_result(end, 3) - food(2)*new_result(end, 4);
                new_result(end, 3) = 0;
                new_result(end, 4) = 0;
                flag = 1;
            else
                new_result(end, 2) = new_result(end, 2) -
                    2*water(2)*new_result(end, 3) - 2*food(2)*new_result(end,
                    4);
                new_result(end, 3) = 0;
                new_result(end, 4) = 0;
            end
        end
        dfs2(new_day, cur_loc, new_result, flag);
    end
elseif cur_loc == 39
    to = [11, 12, 13, 14];
    for i = 1:4
        new_result = result;
        new_day = cur_day;
        temp_route = route{to(i)};
        cnt = 1;
        while cnt <= length(route{to(i)})
            new_day = new_day + 1;
            if new_day > 30 % 超出天30
                status = 0;
                return;
            end
            if days(new_day) ~= 3 % 不是沙暴天气
                cur_loc = temp_route(cnt);
                cnt = cnt + 1;
                new_result(new_day+1, 1) = cur_loc; % 当前的位置
                new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                    前一天相同，矿山第一天不能挣钱
                new_result(new_day+1, 3) = new_result(new_day, 3) +
                    2*water(days(new_day)+2);
                new_result(new_day+1, 4) = new_result(new_day, 4) +
                    2*food(days(new_day)+2);
```

```
        else
            cur_loc = cur_loc;
            new_result(new_day+1, 1) = cur_loc;
            new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                前一天相同，矿山第一天不能挣钱
            new_result(new_day+1, 3) = new_result(new_day, 3) +
                water(days(new_day)+2);
            new_result(new_day+1, 4) = new_result(new_day, 4) +
                food(days(new_day)+2);
        end
    end
    % 判断是否失败
    if new_day > 30
        status = 0;
        continue;
    end
    if new_day == 30 && new_result(end, 1) ~= 64
        status = 0;
        continue;
    end
    if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
        status = 0;
        continue;
    end
    % 触发事件到达商店，清零吃掉的水和食物
    if new_result(end, 1) == 39 || new_result(end, 1) == 62
        if flag == 0 % 如果是第一次来村庄
            new_result(end, 2) = new_result(end, 2) -
                water(2)*new_result(end, 3) - food(2)*new_result(end, 4);
            new_result(end, 3) = 0;
            new_result(end, 4) = 0;
            flag = 1;
        else
            new_result(end, 2) = new_result(end, 2) -
                2*water(2)*new_result(end, 3) - 2*food(2)*new_result(end,
                4);
            new_result(end, 3) = 0;
            new_result(end, 4) = 0;
```

```
            end
        end
        dfs2(new_day, cur_loc, new_result, flag);
    end
elseif cur_loc == 62
    to = [19, 20, 21, 22];
    for i = 1:4
        new_result = result;
        new_day = cur_day;
        temp_route = route{to(i)};
        cnt = 1;
        while cnt <= length(route{to(i)})
            new_day = new_day + 1;
            if new_day > 30 % 超出天30
                status = 0;
                return;
            end
            if days(new_day) ~= 3 % 不是沙暴天气
                cur_loc = temp_route(cnt);
                cnt = cnt + 1;
                new_result(new_day+1, 1) = cur_loc; % 当前的位置
                new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                    前一天相同，矿山第一天不能挣钱
                new_result(new_day+1, 3) = new_result(new_day, 3) +
                    2*water(days(new_day)+2);
                new_result(new_day+1, 4) = new_result(new_day, 4) +
                    2*food(days(new_day)+2);
            else
                cur_loc = cur_loc;
                new_result(new_day+1, 1) = cur_loc;
                new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                    前一天相同，矿山第一天不能挣钱
                new_result(new_day+1, 3) = new_result(new_day, 3) +
                    water(days(new_day)+2);
                new_result(new_day+1, 4) = new_result(new_day, 4) +
                    food(days(new_day)+2);
            end
        end
```

```
        % 判断是否失败
        if new_day > 30
            status = 0;
            continue;
        end
        if new_day == 30 && new_result(end, 1) ~= 64
            status = 0;
            continue;
        end
        if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
            status = 0;
            continue;
        end
        % 触发事件到达商店，清零吃掉的水和食物
        if new_result(end, 1) == 39 || new_result(end, 1) == 62
            if flag == 0 % 如果是第一次来村庄
                new_result(end, 2) = new_result(end, 2) -
                    water(2)*new_result(end, 3) - food(2)*new_result(end, 4);
                new_result(end, 3) = 0;
                new_result(end, 4) = 0;
                flag = 1;
            else
                new_result(end, 2) = new_result(end, 2) -
                    2*water(2)*new_result(end, 3) - 2*food(2)*new_result(end,
                    4);
                new_result(end, 3) = 0;
                new_result(end, 4) = 0;
            end
        end
        dfs2(new_day, cur_loc, new_result, flag);
    end
elseif cur_loc == 30
    to = [5, 8, 9, 10, 6, 7];
    % 在矿山挖矿
    for i = 5:5
        new_result = result;
        new_day = cur_day;
```

```
        temp_route = route{to(i)};
        new_day = new_day + 1;
        if new_day > 30 % 超出天30
            status = 0;
            return;
        end
        % 不管是不是沙尘暴，我都要挖矿
        cur_loc = temp_route(1);
        new_result(new_day+1, 1) = cur_loc;
        new_result(new_day+1, 2) = new_result(new_day, 2) + 1000;
        new_result(new_day+1, 3) = new_result(new_day, 3) +
            3*water(days(new_day)+2);
        new_result(new_day+1, 4) = new_result(new_day, 4) +
            3*food(days(new_day)+2);
        % 判断是否失败
        if new_day > 30
            status = 0;
            continue;
        end
        if new_day == 30 && new_result(end, 1) ~= 27
            status = 0;
            continue;
        end
        if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
            status = 0;
            continue;
        end
        dfs2(new_day, cur_loc, new_result, flag);
    end
    % 在矿山不挖矿
    for i = 6:6
        new_result = result;
        new_day = cur_day;
        temp_route = route{to(i)};
        new_day = new_day + 1;
        if new_day > 30 % 超出天30
            status = 0;
```

```
            return;
        end
        % 不挖矿，只休息
        cur_loc = temp_route(1);
        new_result(new_day+1, 1) = cur_loc;
        new_result(new_day+1, 2) = new_result(new_day, 2);
        new_result(new_day+1, 3) = new_result(new_day, 3) +
            water(days(new_day)+2);
        new_result(new_day+1, 4) = new_result(new_day, 4) +
            food(days(new_day)+2);
        % 判断是否失败
        if new_day > 30
            status = 0;
            continue;
        end
        if new_day == 30 && new_result(end, 1) ~= 64
            status = 0;
            continue;
        end
        if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
            status = 0;
            continue;
        end
        dfs2(new_day, cur_loc, new_result, flag);
    end
    for i = 1:4
        new_result = result;
        new_day = cur_day;
        temp_route = route{to(i)};
        cnt = 1;
        while cnt <= length(route{to(i)})
            new_day = new_day + 1;
            if new_day > 30 % 超出天30
                status = 0;
                return;
            end
            if days(new_day) ~= 3 % 不是沙暴天气
```

```
            cur_loc = temp_route(cnt);
            cnt = cnt + 1;
            new_result(new_day+1, 1) = cur_loc; % 当前的位置
            new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                前一天相同，矿山第一天不能挣钱
            new_result(new_day+1, 3) = new_result(new_day, 3) +
                2*water(days(new_day)+2);
            new_result(new_day+1, 4) = new_result(new_day, 4) +
                2*food(days(new_day)+2);
        else
            cur_loc = cur_loc;
            new_result(new_day+1, 1) = cur_loc;
            new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                前一天相同，矿山第一天不能挣钱
            new_result(new_day+1, 3) = new_result(new_day, 3) +
                water(days(new_day)+2);
            new_result(new_day+1, 4) = new_result(new_day, 4) +
                food(days(new_day)+2);
        end
    end
    % 判断是否失败
    if new_day > 30
        status = 0;
        continue;
    end
    if new_day == 30 && new_result(end, 1) ~= 64
        status = 0;
        continue;
    end
    if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
        status = 0;
        continue;
    end
    % 触发事件到达商店，清零吃掉的水和食物
    if new_result(end, 1) == 39 || new_result(end, 1) == 62
        if flag == 0 % 如果是第一次来村庄
            new_result(end, 2) = new_result(end, 2) -
                water(2)*new_result(end, 3) - food(2)*new_result(end, 4);
```

```
                new_result(end, 3) = 0;
                new_result(end, 4) = 0;
                flag = 1;
            else
                new_result(end, 2) = new_result(end, 2) -
                    2*water(2)*new_result(end, 3) - 2*food(2)*new_result(end,
                    4);
                new_result(end, 3) = 0;
                new_result(end, 4) = 0;
            end
        end
        dfs2(new_day, cur_loc, new_result, flag);
    end
elseif cur_loc == 55
    to = [15, 16, 17, 18, 23, 24];
    % 在矿山挖矿
    for i = 5:5
        new_result = result;
        new_day = cur_day;
        temp_route = route{to(i)};
        new_day = new_day + 1;
        if new_day > 30 % 超出天30
            status = 0;
            return;
        end
        % 不管是不是沙尘暴，我都要挖矿
        cur_loc = temp_route(1);
        new_result(new_day+1, 1) = cur_loc;
        new_result(new_day+1, 2) = new_result(new_day, 2) + 1000;
        new_result(new_day+1, 3) = new_result(new_day, 3) +
            3*water(days(new_day)+2);
        new_result(new_day+1, 4) = new_result(new_day, 4) +
            3*food(days(new_day)+2);
        % 判断是否失败
        if new_day > 30
            status = 0;
            continue;
```

```
        end
        if new_day == 30 && new_result(end, 1) ~= 64
            status = 0;
            continue;
        end
        if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
            status = 0;
            continue;
        end
        dfs2(new_day, cur_loc, new_result, flag);
    end
    % 在矿山不挖矿
    for i = 6:6
        new_result = result;
        new_day = cur_day;
        temp_route = route{to(i)};
        new_day = new_day + 1;
        if new_day > 30 % 超出天30
            status = 0;
            return;
        end
        % 不挖矿，只休息
        cur_loc = temp_route(1);
        new_result(new_day+1, 1) = cur_loc;
        new_result(new_day+1, 2) = new_result(new_day, 2);
        new_result(new_day+1, 3) = new_result(new_day, 3) +
            water(days(new_day)+2);
        new_result(new_day+1, 4) = new_result(new_day, 4) +
            food(days(new_day)+2);
        % 判断是否失败
        if new_day > 30
            status = 0;
            continue;
        end
        if new_day == 30 && new_result(end, 1) ~= 27
            status = 0;
            continue;
```

```
        end
        if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
            status = 0;
            continue;
        end
        dfs2(new_day, cur_loc, new_result, flag);
    end
    for i = 1:4
        new_result = result;
        new_day = cur_day;
        temp_route = route{to(i)};
        cnt = 1;
        while cnt <= length(route{to(i)})
            new_day = new_day + 1;
            if new_day > 30 % 超出天30
                status = 0;
                return;
            end
            if days(new_day) ~= 3 % 不是沙暴天气
                cur_loc = temp_route(cnt);
                cnt = cnt + 1;
                new_result(new_day+1, 1) = cur_loc; % 当前的位置
                new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                    前一天相同，矿山第一天不能挣钱
                new_result(new_day+1, 3) = new_result(new_day, 3) +
                    2*water(days(new_day)+2);
                new_result(new_day+1, 4) = new_result(new_day, 4) +
                    2*food(days(new_day)+2);
            else
                cur_loc = cur_loc;
                new_result(new_day+1, 1) = cur_loc;
                new_result(new_day+1, 2) = new_result(new_day, 2); % 赚的钱和
                    前一天相同，矿山第一天不能挣钱
                new_result(new_day+1, 3) = new_result(new_day, 3) +
                    water(days(new_day)+2);
                new_result(new_day+1, 4) = new_result(new_day, 4) +
                    food(days(new_day)+2);
            end
```

```
        end
        % 判断是否失败
        if new_day > 30
            status = 0;
            continue;
        end
        if new_day == 30 && new_result(end, 1) ~= 64
            status = 0;
            continue;
        end
        if water(1)*new_result(end,3)+food(1)*new_result(end,4) > 1200
            status = 0;
            continue;
        end
        % 触发事件到达商店，清零吃掉的水和食物
        if new_result(end, 1) == 39 || new_result(end, 1) == 62
            if flag == 0 % 如果是第一次来村庄
                new_result(end, 2) = new_result(end, 2) -
                    water(2)*new_result(end, 3) - food(2)*new_result(end, 4);
                new_result(end, 3) = 0;
                new_result(end, 4) = 0;
                flag = 1;
            else
                new_result(end, 2) = new_result(end, 2) -
                    2*water(2)*new_result(end, 3) - 2*food(2)*new_result(end,
                    4);
                new_result(end, 3) = 0;
                new_result(end, 4) = 0;
            end
        end
        dfs2(new_day, cur_loc, new_result, flag);
    end
else % 到达了终点
    final_money = result(end, 2) - 2*water(2)*result(end, 3) -
        2*food(2)*result(end, 4);
%     if check == 0
%         result
```

```
%         check = 1;
%     end
    final_money;
    result(end, 2) = final_money;
    if final_money > max_money
        final_results = result;
        max_money = final_money;
        max_money;
    end
    return;
end
```

1.6 p6main

```
clear, clc;
global m;
m = ones(25, 25);
m = m * 10000;
m(1, [2,6]) = 1;
m(5, [4,10]) = 1;
m(21, [16, 22]) = 1;
m(25, [20, 24]) = 1;
for i = 2:4
    m(i, [i-1,i+1,i+5]) = 1;
end
for i = 22:24
    m(i, [i-5,i-1,i+1]) = 1;
end
for i = 6:5:16
    m(i, [i-5, i+5, i+1]) = 1;
end
for i = 10:5:20
    m(i, [i-5,i+5,i-1])=1;
end
xxx = [7,8,9,12,13,14,17,18,19];
for i = 1:length(xxx)
```

```
    m(xxx(i), [xxx(i)-5,xxx(i)-1,xxx(i)+1,xxx(i)+5]) = 1;
end

for i = 1:25
    m(i, i) = 0;
end
for k = 1:25
    for i = 1:25
        for j = 1:25
            if m(i,k)+m(k,j) < m(i,j)
                m(i, j) = m(i, k) + m(k, j);
            end
        end
    end
end


failure_cnt1 = 0;
failure_cnt2 = 0;
failure_cnt3 = 0;
failure_cnt4 = 0;
failure_cnt5 = 0;
failure_cnt6 = 0;
failure_cnt7 = 0;
failure_cnt8 = 0;
failure_cnt9 = 0;
success_cnt = 0;
max_money = 0;

%% 开始模拟
for kkk = 1:100000
    flag1 = 0; % 是否是初始点购买物资
    flag = 0;
    water = [3, 5, 3, 9, 10];
    food = [2, 10, 4, 9, 10];
    for i = 1:30
        weather_p = rand;
```

```
        if weather_p > 0 && weather_p < 0.3
            weather(i) = 1;
        elseif weather_p >= 0.3 && weather_p < 0.9
            weather(i) = 2;
        else
            weather(i) = 3;
        end
    end
    % 第天0
    result1 = zeros(1, 4);
    result1(1, 1) = 1;
    result1(1, 2) = 10000;
    result1(1, 3) = 0;
    result1(1, 4) = 0;
    result2 = zeros(1, 4);
    result2(1, 1) = 1;
    result2(1, 2) = 10000;
    result2(1, 3) = 0;
    result2(1, 4) = 0;
    result3 = zeros(1, 4);
    result3(1, 1) = 1;
    result3(1, 2) = 10000;
    result3(1, 3) = 0;
    result3(1, 4) = 0;
    cur_loc1 = 1;
    cur_loc2 = 1;
    cur_loc3 = 1;

    cur_day = 0;

    for i = 1:30
        cur_day = cur_day + 1;
        cur_weather = weather(i);
        % 判断天气
        if cur_weather == 3
            % 一个人一个人来决策
            % 第一个人进行决策
```

```
            cur_loc1 = cur_loc1;
            cur_loc2 = cur_loc2;
            cur_loc3 = cur_loc3;
            if cur_loc1 == 14 % 如果在村庄，那么先吃饭，后补给
                result1(cur_day+1, 1) = cur_loc1;
                result1(cur_day+1, 3) = result1(cur_day, 3) +
                    water(cur_weather+2);
                result1(cur_day+1, 4) = result1(cur_day, 4) +
                    food(cur_weather+2);
                % TODO判断这个时候还有没有东西吃:
                if water(1)*result1(cur_day+1, 3) +
                    food(1)*result1(cur_day+1,4) > 1200
                    status = 0;
                    failure_cnt1 = failure_cnt1 + 1;
                    flag = 1;
                    break;
                end
                result1(cur_day+1, 2) = result1(cur_day, 2) -
                    2*water(2)*result1(cur_day+1, 3) -
                    2*food(2)*result1(cur_day+1,4);
                result1(cur_day+1, 3) = 0;
                result1(cur_day+1, 4) = 0;
            elseif cur_loc1 == 18 % 如果被困在矿山，不挖矿
                result1(cur_day+1, 1) = cur_loc1;
                result1(cur_day+1, 2) = result1(cur_day, 2);
                result1(cur_day+1, 3) = result1(cur_day, 3) +
                    water(cur_weather+2);
                result1(cur_day+1, 4) = result1(cur_day, 4) +
                    food(cur_weather+2);
                % TODO判断这个时候有没有东西吃:
                if water(1)*result1(cur_day+1, 3) +
                    food(1)*result1(cur_day+1,4) > 1200
                    status = 0;
                    failure_cnt2 = failure_cnt2 + 1;
                    flag = 1;
                    break;
                end
```

```
        else % 在路上被困住了
            result1(cur_day+1, 1) = cur_loc1;
            result1(cur_day+1, 2) = result1(cur_day, 2);
            result1(cur_day+1, 3) = result1(cur_day, 3) +
                water(cur_weather+2);
            result1(cur_day+1, 4) = result1(cur_day, 4) +
                food(cur_weather+2);
            % TODO判断这个时候有没有东西吃:
            if water(1)*result1(cur_day+1, 3) +
                food(1)*result1(cur_day+1,4) > 1200
                status = 0;
                failure_cnt3 = failure_cnt3 + 1;
                flag = 1;
                break;
            end
        end
        if cur_loc2 == 14 % 如果在村庄, 那么先吃饭, 后补给
            result2(cur_day+1, 1) = cur_loc2;
            result2(cur_day+1, 3) = result2(cur_day, 3) +
                water(cur_weather+2);
            result2(cur_day+1, 4) = result2(cur_day, 4) +
                food(cur_weather+2);
            % TODO判断这个时候还有没有东西吃:
            if water(1)*result2(cur_day+1, 3) +
                food(1)*result2(cur_day+1,4) > 1200
                status = 0;
                failure_cnt1 = failure_cnt1 + 1;
                flag = 1;
                break;
            end
            result2(cur_day+1, 2) = result2(cur_day, 2) -
                2*water(2)*result2(cur_day+1, 3) -
                2*food(2)*result2(cur_day+1,4);
            result2(cur_day+1, 3) = 0;
            result2(cur_day+1, 4) = 0;
        elseif cur_loc2 == 18 % 如果被困在矿山, 不挖矿
            result2(cur_day+1, 1) = cur_loc2;
```

```
            result2(cur_day+1, 2) = result2(cur_day, 2);
            result2(cur_day+1, 3) = result2(cur_day, 3) +
                water(cur_weather+2);
            result2(cur_day+1, 4) = result2(cur_day, 4) +
                food(cur_weather+2);
            % TODO判断这个时候有没有东西吃:
            if water(1)*result1(cur_day+1, 3) +
                food(1)*result1(cur_day+1,4) > 1200
                status = 0;
                failure_cnt2 = failure_cnt2 + 1;
                flag = 1;
                break;
            end
        else % 在路上被困住了
            result1(cur_day+1, 1) = cur_loc1;
            result1(cur_day+1, 2) = result1(cur_day, 2);
            result1(cur_day+1, 3) = result1(cur_day, 3) +
                water(cur_weather+2);
            result1(cur_day+1, 4) = result1(cur_day, 4) +
                food(cur_weather+2);
            % TODO判断这个时候有没有东西吃:
            if water(1)*result1(cur_day+1, 3) +
                food(1)*result1(cur_day+1,4) > 1200
                status = 0;
                failure_cnt3 = failure_cnt3 + 1;
                flag = 1;
                break;
            end
        end
    else % 不是沙暴天气
        available_places = find(m(cur_loc, :) == 1);
        available_places = [available_places, cur_loc];
        values = [];
        for k = 1:length(available_places)
            % 下面这一行可能有点问题1
            values(k) = compute_value(available_places(k), cur_day,
                result(cur_day, 3), result(cur_day, 4));
```

```
        end
        [x, y] = max(values); % 可能有多个y
        y = available_places(y(1)); % 选择第一个, 之后可以换成随机选择一个
        result(cur_day+1, 1) = y; % 移动
        cur_loc = y;
        if result(cur_day+1, 1) == 14 % 如果移动到村庄
            result(cur_day+1, 3) = result(cur_day, 3) +
                2*water(cur_weather+2);
            result(cur_day+1, 4) = result(cur_day, 4) +
                2*food(cur_weather+2);
            if water(1)*result(cur_day+1, 3) +
                food(1)*result(cur_day+1,4) > 1200
                status = 0;
                failure_cnt4 = failure_cnt4 + 1;
                flag = 1;
                break;
            end
            if flag1 == 0
                result(cur_day+1, 2) = result(cur_day, 2) -
                    water(2)*result(cur_day+1, 3) -
                    food(2)*result(cur_day+1,4);
                flag1 = 1;
            else
                result(cur_day+1, 2) = result(cur_day, 2) -
                    2*water(2)*result(cur_day+1, 3) -
                    2*food(2)*result(cur_day+1,4);
            end
            %result(cur_day+1, 2) = result(cur_day, 2) -
                2*water(2)*result(cur_day+1, 3) -
                2*food(2)*result(cur_day+1,4);
            result(cur_day+1, 3) = 0;
            result(cur_day+1, 4) = 0;
        elseif result(cur_day+1, 1) == 18 && result(cur_day, 1) ~= 18% 如
            果移动到矿山, 第一天无法挖矿
            result(cur_day+1, 3) = result(cur_day, 3) +
                2*water(cur_weather+2);
            result(cur_day+1, 4) = result(cur_day, 4) +
                2*food(cur_weather+2);
```

```
            result(cur_day+1, 2) = result(cur_day, 2);
            if water(1)*result(cur_day+1, 3) +
                food(1)*result(cur_day+1,4) > 1200
                status = 0;
                failure_cnt5 = failure_cnt5 + 1;
                flag = 1;
                break;
            end
        elseif result(cur_day+1, 1) == 18 && result(cur_day, 1) == 18 %
            从矿山到矿山，那么是挖
            矿
            result(cur_day+1, 2) = result(cur_day, 2) + 1000;
            result(cur_day+1, 3) = result(cur_day, 3) +
                3*water(cur_weather+2);
            result(cur_day+1, 4) = result(cur_day, 4) +
                3*food(cur_weather+2);
            if water(1)*result(cur_day+1, 3) +
                food(1)*result(cur_day+1,4) > 1200
                status = 0;
                failure_cnt6 = failure_cnt6 + 1;
                flag = 1;
                break;
            end
        elseif result(cur_day+1, 1) == 25 % 如果到达终点
            status = 1;
            result(cur_day+1, 3) = result(cur_day, 3) +
                2*water(cur_weather+2);
            result(cur_day+1, 4) = result(cur_day, 4) +
                2*food(cur_weather+2);
            if water(1)*result(cur_day+1, 3) +
                food(1)*result(cur_day+1,4) > 1200
                status = 0;
                failure_cnt7 = failure_cnt7 + 1;
                flag = 1;
                break;
            end
            result(cur_day+1, 2) = result(cur_day, 2) -
                2*water(2)*result(cur_day+1, 3) -
```

```
                2*food(2)*result(cur_day+1, 4);
            result(cur_day+1, 4) = 0;
            result(cur_day+1, 3) = 0;
            if result(cur_day+1, 2) > max_money
                max_money = result(cur_day+1, 2);
                final_result = [];
                final_result = result;
            end
            success_cnt = success_cnt + 1;
            earn(success_cnt) = result(cur_day+1, 2);
            flag = 1;
            break;
        else % 如果不是特殊点
            result(cur_day+1, 3) = result(cur_day, 3) +
                2*water(cur_weather+2);
            result(cur_day+1, 4) = result(cur_day, 4) +
                2*food(cur_weather+2);
            result(cur_day+1, 2) = result(cur_day, 2);
            if water(1)*result(cur_day+1, 3) +
                food(1)*result(cur_day+1,4) > 1200
                status = 0;
                failure_cnt8 = failure_cnt8 + 1;
                flag = 1;
                break;
            end
        end
    end
    if flag == 0
        failure_cnt9 = failure_cnt9 + 1;
        bad_results{failure_cnt9} = result;
    end
end
hist(earn);
mean(earn)
```

1.7 ImperialistCompetitveAlgorithm,GlobalOptimizationStrategy

```
close all
clc; clear
global s;

%% Problem Statement
ProblemParams.CostFuncName = 'rbtestsinfunc';
ProblemParams.CostFuncExtraParams = '';
ProblemParams.NPar = 3;
ProblemParams.VarMin = -1;
ProblemParams.VarMax = 2;

if numel(ProblemParams.VarMin)==1
    ProblemParams.VarMin=repmat(ProblemParams.VarMin,1,ProblemParams.NPar);
    ProblemParams.VarMax=repmat(ProblemParams.VarMax,1,ProblemParams.NPar);
end

ProblemParams.SearchSpaceSize = ProblemParams.VarMax -
    ProblemParams.VarMin;

s = [];
%% Algorithmic Parameter Setting
AlgorithmParams.NumOfCountries = 200;
AlgorithmParams.NumOfInitialImperialists = 8;
AlgorithmParams.NumOfAllColonies = AlgorithmParams.NumOfCountries -
    AlgorithmParams.NumOfInitialImperialists;
AlgorithmParams.NumOfDecades = 2000;
AlgorithmParams.RevolutionRate = 0.3;
AlgorithmParams.AssimilationCoefficient = 2;
AlgorithmParams.AssimilationAngleCoefficient = .5;
AlgorithmParams.Zeta = 0.02;
AlgorithmParams.DampRatio = 0.99;
AlgorithmParams.StopIfJustOneEmpire = false;
AlgorithmParams.UnitingThreshold = 0.02;

zarib = 1.05;
```

```
alpha = 0.1;

%% Display Setting
DisplayParams.PlotEmpires = false;
if DisplayParams.PlotEmpires
    DisplayParams.EmpiresFigureHandle = figure('Name','Plot of
        Empires','NumberTitle','off');
    DisplayParams.EmpiresAxisHandle = axes;
end

DisplayParams.PlotCost = true; % "true" to plot. "false"
if DisplayParams.PlotCost
    DisplayParams.CostFigureHandle = figure('Name','Plot of Minimum and
        Mean Costs','NumberTitle','off');
    DisplayParams.CostAxisHandle = axes;
end

ColorMatrix = [1 0  0  ; 0 1  0   ; 0  0 1   ; 1  1  0 ; 1  0 1   ; 0 1  1
      ; 1 1 1     ;
             0.5 0.5 0.5; 0 0.5 0.5 ; 0.5 0 0.5 ; 0.5 0.5 0 ; 0.5 0 0 ; 0
                 0.5 0 ; 0 0 0.5 ;
             1   0.5 1 ; 0.1*[1 1 1]; 0.2*[1 1 1]; 0.3*[1 1 1]; 0.4*[1 1
                 1]; 0.5*[1 1 1]; 0.6*[1 1 1]];
DisplayParams.ColorMatrix = [ColorMatrix ; sqrt(ColorMatrix)];

DisplayParams.AxisMargin.Min = ProblemParams.VarMin;
DisplayParams.AxisMargin.Max = ProblemParams.VarMax;

%% Creation of Initial Empires
InitialCountries = GenerateNewCountry(AlgorithmParams.NumOfCountries ,
    ProblemParams);

% Calculates the cost of each country. The less the cost is, the more is
    the power.
if isempty(ProblemParams.CostFuncExtraParams)
    InitialCost = feval(ProblemParams.CostFuncName,InitialCountries);
else
```

```
    InitialCost =
        feval(ProblemParams.CostFuncName,InitialCountries,ProblemParams.CostFuncExtraPara
end
[InitialCost,SortInd] = sort(InitialCost);                % Sort the cost
    in assending order. The best countries will be in higher places
InitialCountries = InitialCountries(SortInd,:);           % Sort the
    population with respect to their cost.

Empires =
    CreateInitialEmpires(InitialCountries,InitialCost,AlgorithmParams,
    ProblemParams);

%% Main Loop
MinimumCost = repmat(nan,AlgorithmParams.NumOfDecades,1);
MeanCost = repmat(nan,AlgorithmParams.NumOfDecades,1);

if DisplayParams.PlotCost
    axes(DisplayParams.CostAxisHandle);
    if any(findall(0)==DisplayParams.CostFigureHandle)
        h_MinCostPlot=plot(MinimumCost,'r','LineWidth',1.5,'YDataSource','MinimumCost');
        hold on;
        h_MeanCostPlot=plot(MeanCost,'k:','LineWidth',1.5,'YDataSource','MeanCost');
        hold off;
        pause(0.05);
    end
end

for Decade = 1:AlgorithmParams.NumOfDecades
    AlgorithmParams.RevolutionRate = AlgorithmParams.DampRatio *
        AlgorithmParams.RevolutionRate;

    Remained = AlgorithmParams.NumOfDecades - Decade
    for ii = 1:numel(Empires)
        %% Assimilation; Movement of Colonies Toward Imperialists
            (Assimilation Policy)
        Empires(ii) =
            AssimilateColonies(Empires(ii),AlgorithmParams,ProblemParams);
```

```
        %% Revolution; A Sudden Change in the Socio-Political
            Characteristics
        % 该殖民地改革操作可能导致势力较强的殖民地丢失，导致寻优精度降低
        Empires(ii) =
            RevolveColonies(Empires(ii),AlgorithmParams,ProblemParams);

        %% New Cost Evaluation
        if isempty(ProblemParams.CostFuncExtraParams)
            Empires(ii).ColoniesCost =
                feval(ProblemParams.CostFuncName,Empires(ii).ColoniesPosition);
        else
            Empires(ii).ColoniesCost =
                feval(ProblemParams.CostFuncName,Empires(ii).ColoniesPosition,ProblemParams
        end

        %% Empire Possession (****** Power Possession, Empire Possession)
        Empires(ii) = PossesEmpire(Empires(ii));

        %% Computation of Total Cost for Empires
        Empires(ii).TotalCost = Empires(ii).ImperialistCost +
            AlgorithmParams.Zeta * mean(Empires(ii).ColoniesCost);

    end

    %% Uniting Similiar Empires
    Empires = UniteSimilarEmpires(Empires,AlgorithmParams,ProblemParams);

    %% Imperialistic Competition
    Empires = ImperialisticCompetition(Empires);

    if numel(Empires) == 1 && AlgorithmParams.StopIfJustOneEmpire
        break
    end

    %% Displaying the Results
    DisplayEmpires(Empires,AlgorithmParams,ProblemParams,DisplayParams);
```

```
    ImerialistCosts = [Empires.ImperialistCost];
    MinimumCost(Decade) = min(ImerialistCosts);
    MeanCost(Decade) = mean(ImerialistCosts);

    if DisplayParams.PlotCost
        refreshdata(h_MinCostPlot);
        refreshdata(h_MeanCostPlot);
        drawnow;
        pause(0.01);
    end

end
MinimumCost(end)
```