**“穿越沙漠”游戏最佳策略模型的研究与构建**

**摘 要**

“穿越沙漠”是一种小游戏，选择最佳的游戏策略，能使参与者在到达终点时保留尽可能多的资金；为帮助参与者选择最优策略，我们使用贪心法、回溯法以求解单目标最优化模型，同时利用基于收益的数学期望的决策函数帮助参与者动态选择后继结点；在多个玩家参与的情况下，我们在博弈论的有关理论的基础上，仍然使用最优化模型或决策函数，成功地帮助每一关的参与者选择了最佳策略。

针对问题一，我们认为参与者不可能在任意结点停留，因此我们首先证明：除了沙暴天气必须停留以外，参与者只可能在矿山停留挖矿，其余地点不做停留；然后，我们再根据游戏规则，给出参与者向前行走、沙暴时原地停留、矿山挖矿、村庄补给等情况下生活必需品以及资金剩余量的**递推关系式**的基础之上，以剩余资金最多为目标，设置了参与者按时到达、生活必需品重量不超过负重上限的约束条件，**建立了单目标最优化模型**，并对参与者挖矿和不挖矿的情况分别使用**贪心法**和**回溯法**，使用**Matlab**和**C++**编程求解，最终得到：在最佳策略下，第一关、第二关的剩余资金分别为10470、12730元，并将相应结果填入了Result.xlsx文件；随后，我们对该模型进行了灵敏度分析。

问题二中，考虑到参与者的决策关键为资金的**数学期望**，因此我们在给出晴朗、高温、沙暴这三种天气的概率的基础之上，得到了每天行走所耗资源对应的金额的期望、挖矿净收益的期望、挖矿所耗资源对应的金额的期望以及挖矿所耗水和食物箱数的期望的表达式；然后，借助这几种期望的比较，我们建立了决定参与者何时直奔终点、前往矿山或村庄、在矿山停留或离开矿山的**决策模型**，并给出了对应的**决策函数**，使用**Matlab**编程可以得到；第三关中，参与者应直奔终点，第四关中，参与者从起点前往村庄，再前往矿山，最终到达终点，且剩余资金为**10065**元；为验证该模型的正确性，我们使用**蒙特卡洛方法**对该模型进行了仿真。

对于问题三的第一部分，我们参考了**博弈论**的有关理论，对有*n*名玩家和两名玩家的情况，构造了相应的支付矩阵和效用期望，在此基础上，建立了基于博弈论的**最佳策略模型**；为使求解步骤更具有一般性，我们首先给出了*n*名玩家的一般情况下的求解过程，然后对于第五关中只有两名玩家的情况，先根据问题一中代码的运行结果，选择了3条较优路线，作为两名玩家选择路线的决策集，接着使用**Lingo**编程，计算出了这两名玩家选择各条路线的概率，由此得出了这种情况下两位玩家选择的路线。

针对问题三的第二部分，考虑到这一问具有问题三第一部分和问题二的共同点，因此我们在使用**博弈论**有关理论计算出各玩家前往各结点的概率基础之上，使用与问题二中类似的分析方法，给出了玩家行走、停留、挖矿、补给的情况下，某结点净收益的**数学期望**表达式，根据该数学期望的大小，即可得到选择下一结点的**决策模型**，并编写了相应的迭代算法，以求解得出3位参赛者选择的路线。

最后，我们分析了模型的优缺点，在对缺点提出相应的改进方式的基础之上，将该模型推广到其他领域。

**关键词：最优化模型、贪心法、回溯法、决策模型、蒙特卡洛方法、博弈论**

# 一、问题重述

有一种“穿越沙漠”小游戏，参与该游戏的玩家在获得沙漠地图，并在起点用初始资金以基准价格购买好水和食物等生活必需品后，从第1天开始在沙漠中行走，其目标是在游戏结束前到达终点，并剩余尽量多的资金；他可以选择在邻接的两个区域之间行走，也可以选择在原地停留；行进的过程中，他可能经过村庄或矿山，在前者可以补给装备，价格为基准价格的两倍，在后者可以挖矿赚取资金；游戏过程中，整个沙漠可能出现晴朗、高温、沙暴这三种天气，它们对游戏参与者是否在某地停留以及参与者的基础消耗量有一定影响。此外，参与者还需要遵守以下游戏规则：

1.参与者携带的生活必需品的重量不能超过他的负重上限，但在到达终点前不能耗尽所有的生活必需品:若他选择行走，则生活必需品的消耗量为原地停留时的两倍。

2.参与者也可在起点停留，或返回起点，但是不能多次在起点补给资源，若在到达终点时生活必需品有剩余，可以将其退回，价格为基准价格的一半。

3.参与者可在矿山挖矿以获得一定的基础收益,但其对生活必需品的消耗量将达到原地停留时的三倍，同时，他在到达矿山的当天不能挖矿,但在出现沙暴天气时可挖矿。

为帮助游戏参与者选择最佳的游戏策略，我们需要建立数学模型完成以下任务：

1.在只有一名玩家，且他知道游戏过程中每天的天气状况的情形下，给出最优策略，并利用第一关、第二关的数据进行求解，并生成相应的Excel文件。

2.在只有一名玩家，但他只知道当天的天气状况的情形下，给出最优策略，并利用第三关、第四关的数据进行求解。

3.在有*n*名玩家，且其中的*k*名玩家可能同时行动的情形下，若游戏过程中每天天气状况已知，且行动方案在第1天前确定不可更改，给出最优策略并对第五关进行讨论。

4.在前一题的条件下，若玩家只知道当天的天气状况以及其他玩家当天的行动方案和剩余资源量，给出最优策略并对第六关进行讨论。

# 二、问题分析

## 2.1 问题一的分析

问题一要求我们在只有一名玩家，且他知道游戏过程中每天的天气状况的情形下，给出最优策略；首先，我们根据题目中给出的游戏规则，可以列出向前行走、沙暴时原地停留、矿山挖矿、村庄补给等情况下生活必需品以及资金剩余量的递推关系式，然后根据参与者在游戏过程中所要满足的各种条件，可以建立单目标最优化模型;考虑到直接沿最短路径行走到终点所需购买的生活必需品数量较少，而在矿山停留挖矿能够赚取更多的资金，因此我们对前一种情况使用弗洛伊德算法，后一种情况使用回溯算法进行求解，以获得剩余资金最多时的游戏策略。

## 2.2 问题二的分析

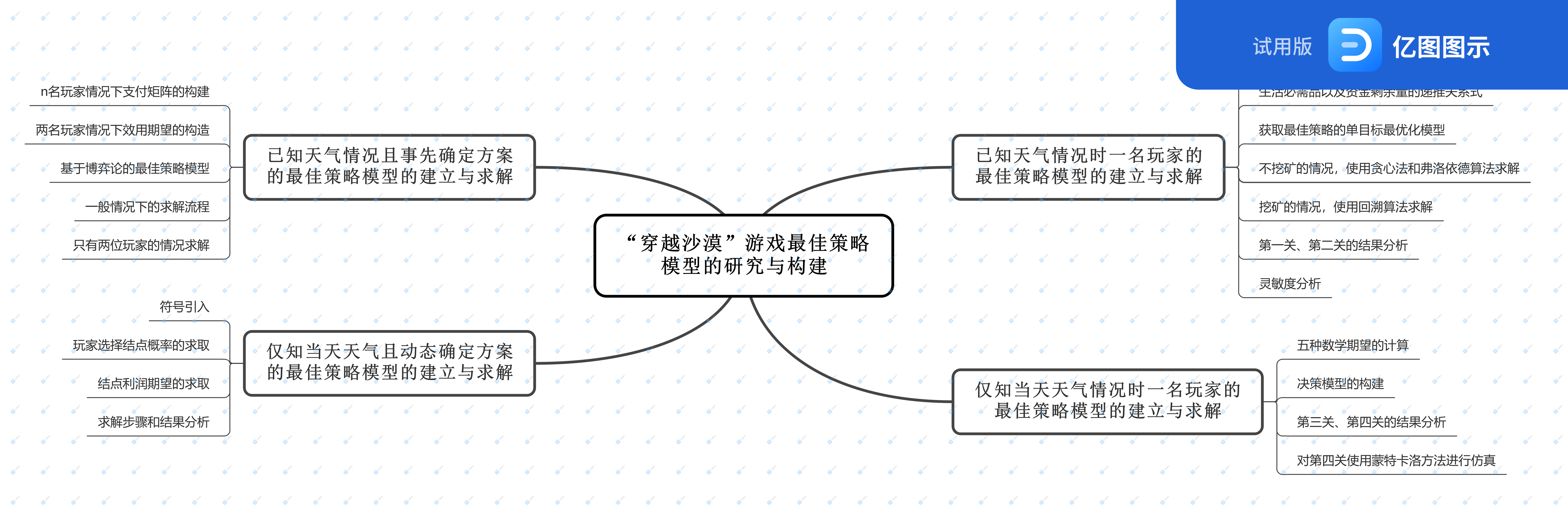
问题二要求我们在只有一名玩家，但他只知道当天的天气状况的情形下，给出最优策略，考虑到参与者的决策方案与每天行走所耗资源对应的金额的期望、挖矿净收益的期望、挖矿所耗资源对应的金额的期望、挖矿所耗水和食物箱数的期望有关，而这五个期望的计算与天气有关，因此我们借助问题一中的天气数据，计算各种天气出现的概率以作为计算这些数学期望的辅助数据，然后根据该思路，我们可以建立基于数学期望的决策模型，并编写相应的迭代算法进行求解以获得参与者的决策方案。

## 2.3 问题三第一部分的分析

问题三第一部分要求我们在有*n*名玩家，且其中的*k*名玩家可能同时行动的情形下，若游戏过程中每天天气状况已知，且行动方案事先确定不可更改，给出最优策略并对第五关进行讨论，由于此问题中有多名玩家，一名玩家的决策方案会与别的玩家有关联，因此我们需要参考博弈论的有关资料，根据题意构建支付矩阵，并在两名玩家的条件下给出效用期望；然后就可以给出一般情况下模型的求解方法，并对本题的情况进行求解。

## **2.4 问题三第二部分的分析**

问题三第二部分将上一部分中部分条件更变为只知道当天的天气状况以及其他玩家当天的行动方案和剩余资源量，给出最优策略并对第六关进行讨论，与问题二类似，我们仍根据参与者前往某个结点所获净收益的期望来判断他前往哪一个结点，而该期望的求解仍需要前一部分使用的博弈论知识以及问题二中定义的部分符号，将这两部分相结合即可得到获得该部分最优策略的数学模型，并可使用迭代算法进行求解。

下面为我们进行问题解决的思维导图。

**图2.4.1：问题解决思维导图**

# 三、模型假设

1.假设沙漠中各天的天气相互独立。

2.假设游戏参与者是完全理性的。

3.假设沙漠中天气不存在突变。

# 四、符号说明

|  |  |  |  |
| --- | --- | --- | --- |
| **关键符号** | **符号说明** | **关键符号** | **符号说明** |
| *T*0 | 到达终点的实际天数 | *BW*(*t*) | 第*t*天水的基础消耗量 |
| *BF*(*t*) | 第*t*天食物的基础消耗量 | *PB* | 参与者挖矿赚的的基础收益 |
| Ⅱ(*x*) | 指示函数 | *E*(*j*) | 第*j*个结点净收益的期望 |
| **关键符号** | **符号说明** | | |
| *W*(*i*,*t*) | 第*t*天参与者在第*i*个结点时水的剩余量 | | |
| *F*(*i*,*t*) | 第*t*天参与者在第*i*个结点时食物的剩余量 | | |
| *R*(*i*,*t*) | 第*t*天参与者在第*i*个结点时资金的剩余量 | | |
| *d*(*i*,*t*) | 从第*i*个结点到第*j*个节点的最短路径 | | |
| *f*(*i*,*t*) | 参与者选择下一个结点的决策函数 | | |

# 五、模型的建立与求解

## 5.1 问题一：已知天气情况时一名玩家的最佳策略模型的与求解

为求解一名玩家在已知天气情况下的最佳策略，我们首先根据游戏规则列出了每天生活必需品以及资金剩余量的递推关系式，然后根据参与者在游戏过程中所要满足的各种条件，建立了单目标最优化模型，并且分别对直接到达终点、在矿山挖矿这两种情况进行求解，以获得最佳策略。

### 5.1.1 已知天气情况时一名玩家的最佳策略模型的建立

* 停留地点的选择

根据游戏规则，参与者可以选择在任何地点停留，但是经过分析，我们认为除了沙暴天气必须停留以外，参与者只有在矿山停留挖矿才能获得收益，在其他地点停留只会带来较大亏损；设*PB*为参与者挖矿获得的基础收益，*PW*、*PF*分别为水和食物的基础价格，下面我们对这两种情况进行具体的分析和说明。

* 矿山挖矿必定获得收益

若参与者在矿山挖矿一天，则这一天他将多消耗一部分水和食物，同时也会晚一天到达终点；我们按照最坏情况进行计算，即他挖矿那一天遭遇了沙暴天气，同时在第17天到达终点前一个结点，根据第一关、第二关的参数设定，此时每天食物和水的基础消耗量均为十箱，同时最后一天其消耗量为基础消耗量的两倍，因此他的收益

，

该式的第二项为相比于正常行进，参与者在沙暴天气挖矿所需多消耗的生活必需品对应的资金；由于参与者在第17天到达终点前一个节点，因此他需要在该节点等待两天，在第三天再前往终点，第三项、第四项即为这三天他所需多消耗的生活必需品对应的资金。经过整理化简，并代入题中所给数据，可得*P*1=350>0，由于该情况为最坏情况，所以我们可以得出结论：参与者挖矿必定获得收益。

* 其它地点停留必定导致亏损

若参与者在其他地点停留，且停留的那天未遭遇沙暴，我们按照最好情况进行计算，即停留的那天以及到达终点的那一天的天气均是晴朗，则其因为停留导致的亏损为

，

该式的第一项为参与者因为在该结点停留所需消耗的生活必需品对应的资金，第二项为参与者晚到终点一天所需多消耗的生活必需品对应的资金；经过整理化简，并代入题中所给数据，可得*R*=285>0，由于该情况为最好情况，所以我们可以得出结论：参与者在其余地点停留必定导致亏损。

* 生活必需品以及资金剩余量的递推关系式

在建立该模型前，我们首先需要给出各个时刻在各个结点处，参与者所带生活必需品以及资金剩余量的递推关系式；设参与者在第*T*0(*T*=1,2,…,*Tmax*)天到达终点，其中*Tmax*为截止时间，终点的编号为*end*、*W*(*i*,*t*)、*F*(*i*,*t*)、*R*(*i*,*t*)分别为第*t*(*t*=0,1，…,*T*0)天参与者在第*i*(*i*=1,2,…, *end*)个结点时水、食物、资金的剩余量，由于若在终点处退回多余的生活必需品，会给参赛者待来一定的损失，因此在最佳策略下，参与者一定在终点处刚好消耗完所有的水和食物，即

 **(5.1.1)**

设*M*0为初始资金，则在起点处购买过物资之后的剩余资金

**,**

* **向前行走时生活必需品剩余量的递推关系式**

设参与者第  天在第  个结点，第  天在第 j个结点, 又设第 t天水的基础消耗量为 , 食物的基础消耗量为 , 由于行走时生活必需 品的消耗量为基础消耗量的两倍，因此



 为  的后继结点的集合，考虑到在前行的过程中，剩余资金量保持不变，所以：



* **沙暴时原地停留时生活必需品剩余量的递推关系式**

在遭遇沙暴天气时，参与者必须原地停留，此时他消耗的生活必需品数量即为基碰 消耗量，因此



在原地停留的情况下，  的表达式与式(5.1.2)相同。

若参与者在矿山挖矿，则其消耗的生活必需品数量为基础消耗量的三倍，因此



在挖矿时，参与者可兼得一定量的基础收益，所以



* **矿山挖矿时生活必需品剩余量的递推关系式**

若诊与者在矿山挖矿，则其消耗的生活必需品数量为基础消耗量的三倍，因此



在挖矿时，参与者可兼得一定量的基础收益，所以



* **村庄补给时生活必需品剩余量的递推关系式**

设参与者在村庄补给的水的数量为  箱，补给的食物的数量为  箱，由于其在村 庄无需停留，所以消耗的生活必需品数量仍为基础消耗量的两倍，因此



S 为 i的后继结点的集合，考虑到在前行的过程中，剩余资金量保持不变，所以



沙暴时原地停留时生活必需品剩余量的递推关系式在遭遇沙暴天气时，参与者必须原地停留，此时他消耗的生活必需品数量即为基克 消耗量，因此



在原地停留的情说下，  的表达式与式(5.1.2)相同。

**6.村庄补给时生活必需品剩余量的递推关系式**

设参与者在村庄补给的水的数量为  箱，补给的食物的数量为  箱，由于其在村 庄无需停留，所以消耗的生活必需品数量仍为基础消耗量的两倍，因此



补给生活必需品需要消耗参与者的资金，且村庄处生活必需品的售价为基准价格的两倍， 所以补给过后的剩余资金



* **获得最佳策略的单目标最优化模型**

为了获得该游戏的最佳策略，我们根据参与者在游戏过程中所要满足的各种条件, 建立了单目标最优化模型。

* **目标函数: 剩余资金最多**

题目中提及，参与者需要保留尽可能多的资金，因此目标函数为：



其中  分别为第  天参与者在第  个结点时资金的剩余 量， end 为终点的编号，  为到达终点的实际时间。

* **约束条件 1: 参与者在截止日期之前到达终点**

根据游戏规则，参与者需要在规定的时间内到达终点，因此一个约束条件为



其中  为到达终点的实际时间，  为该关卡规定的到达终点的截止时间。

+ 约束条件 2: 生活必需品的数量不能超过负重上限

设参与者的载重上限为  分别每箱水和食物的重量，则另一个约束条 件为



其中  、  分别为第  天参与者在第  个结点时水、食物的剩余量。 此外，参与者在到达终点之前不能耗尽所有的生活必需品，因此这也是一个约束条

件，由于我们在给出生活必需品以及资金剩余量的递推关系式之前，已经用式(5.1.1)限 定在最后一个时刻的食物和水的剩余量为 0, 因此在中间过程中不会出现生活必需品已 经耗尽的情况，即我们可以用式(5.1.1)作为该约束条件。

止 获得最佳策略的单目标最优化模型

根据上述分析，我们可以得到获得最佳策略的单目标最优化模型:



其中  分别为第  天参与者在第  个结点时资金的剩余量， end 为终点的编号,  为到达终点的实际时间。

约束条件 1: 参与者在截止日期之前到达终点

根据游戏规则，参与者需要在规定的时间内到达终点，因此一个约束条件为



其中  为到达终点的实际时间，  为该关卡规定的到达终点的截止时间。

* **约束条件 2: 生活必需品的数量不能超过负重上限**

设参与者的载重上限为  、  分别每箱水和食物的重量，则另一个约束条 件为



其中  、  分别为第  天参与者在第  个结点时水、食物的剩余量。 此外，参与者在到达终点之前不能耗尽所有的生活必需品，因此这也是一个约束条 件，由于我们在给出生活必需品以及资金剩余量的递推关系式之前，已经用式(5.1.1)限 定在最后一个时刻的食物和水的剩余量为 0 ，因此在中间过程中不会出现生活必需品已 经耗尽的情况，即我们可以用式(5.1.1)作为该约束条件。

获得最佳策略的单目标最优化模型

根据上述分析，我们可以得到获得**最佳策略的单目标最优化模型**:



设计相应的算法求解该模型即可得到已知天气情况时一名玩家的最佳策略。

### 5.1.2 已知天气情况时一名玩家的最佳模型的求解

不挖矿的情况，使用贪心法和弗洛伊德算法求解,应使用贪心法进行求解: 贪心法是一种重要的算法设计思想，其基本原则是每一次做出 的选择都是在当前情况下看起来最好的选择, 因此依据贪心法设计的算法得到的是某种 意义下的局部最优解，而不一定是全局最优解[2] 准则的选择。设不挖矿时参与者的最大收益为 , 挖矿时参与者的最大收益为 , 则最佳策略下的收益:



**不挖矿的情况下基于贪心准则的分析：**

1.在起点购买尽量少的生活必需品

对于不挖矿时在起点购买物质，我们认为需要购买尽量少的生活必需品，设在起点购买水和食物各  箱，此时恰好能保证参与者到达终点，则资金剩余量：



品，即购买水  箱，食物  箱，则多余的生活必需品需要在终点退回，由于 退回后返回给参与者的费用仅为基础价格的一半，所以



因此若不挖矿时想使得资金剩余量最大，必须在起点购买尽量少的生活必需品。

2.选择最短路径到达终点

若玩家选择不挖矿，则其必须减少生活必需品的消耗，因此必须选择从起点到终点的最短路径，定义函数：



则可以给出第一关、第二关中第  天水和食物的基础消耗量为  的表达式:



设从起点到终点的最短路程为 , 则从起点到终点所需的天数



设从起点到终点的最短路程为 , 则从起点到终点所需的天数



其中  为指示函数[1]，当 x 成立时， , 否则 , 因此上式反映了从起点 到终点所需的天数由选择最短路径行走的天数和沙暴天气停留的天数两部分组成; 因此参与者在起点购买的水和食物的箱数分别为:



到达终点时的资金剩余量:



**求解最佳策略的弗洛伊德算法**

由于在不挖矿时我们直接选择最短路径到达终点，所以需要采用弗洛伊德算法进行求解，具体步骤如下：

1.使用邻接矩阵，将题目中所给的地图转换为图论中的图，并对距离矩阵进行初始化，能直接到达的结点之间距离为 1，否则为无穷大。

2.借助弗洛伊德算法求解任意两点之间的最短距离，即对于任意两个结点，若经过另外某个结点"中转"，得到的距离小于原来两结点的距离，则该距离为两点之间更短的距离;考虑将所有结点作为中转点的情况，即可获得两结点之间的最短路径。

3.初始化天气情况、各种情况下生活必需品的消耗量、最大负重等常量。

4.根据第2步中的结果，我们可以得到从起点到终点的最短路程d，根据式（5.1.3）可算出参与者在起点所需购买的生活必需品数量，进而可根据式（5.1.4）算出到达终点时的资金剩余量;使用 Matlab 编程[3]，可获得相关数据如下表所示。

**表5.1.2.1 不挖矿时参与者在起点购买生活必需品数量及到达时资金剩余量**

|  |  |  |
| --- | --- | --- |
| 关卡编号 | 第一关 | 第二关 |
| 最短路程 | 3 | 11 |
| 在起点购买食物/箱 | 38 | 170 |
| 在起点购买水/箱 | 42 | 182 |
| 到达终点资金剩余量/元 | 9140 | 7390 |
| 最短路径 | 1→25→26→27 | 1→2→3→4→5→13→22→30→39→47→56→64 |

**回溯法求解挖矿策略**

如果玩家选择挖矿可能有多种路线选择方式，例如先前往村庄还矿山，甚至可能来回往返村庄、矿山之间以补给资源，因此，我们必须考虑所有可能的情况，并对一些不可能出现的情况进行剪枝，以减少遍历次数，因此我们设计了回溯算法对挖矿的情况进行求解。

回溯算法适用于解决复杂的路线问题，为了保证结果的准确性我们设计的算法没有按照常规的路径去考察在村庄是否需要补给，而是在每一个结点考察是否需要补给，如果不需要，则继续深度优先遍历，否则回溯到上一个村庄的位置进行补给，或增加在起点购买的生活必需品的数量，以达到剪枝的目的。

算法的具体步骤如下：

1. 将结点分为起点、矿山、村庄、终点四类，并初始化各个常量，如各结点的通达情况数组、距离数组、天气情况、生活必需品的重量、基准价格、基础消耗等，令i = 0。

2.检查i是否已经达到参与者的负重上限M，即 1200千克，若是输出结果，算法结束，否则将i+1赋值给i。

3.由于每千克水的质量为3 千克/箱，食物的质量为2千克/箱，因此令参与者在

M- M-1箱水，起点购买-箱1箱食物，若没有考察过的情况，则回到第四步进行深度优先遍历，否则回到第二步。

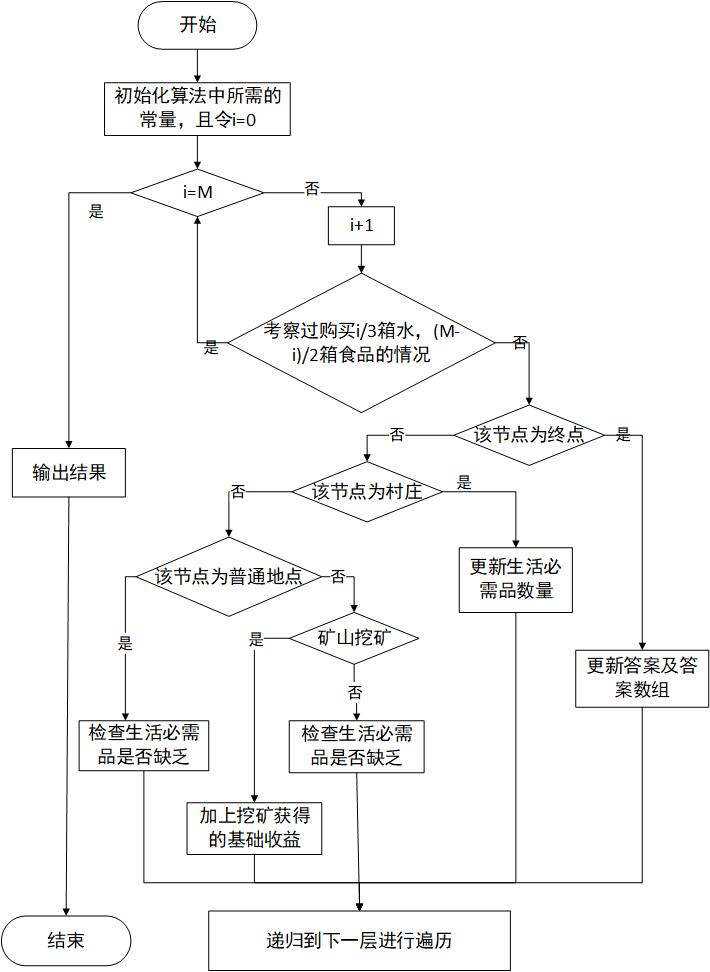
4.若当前情况下，该结点为终点，更新答案及相应的答案数组;若此次遍历时间已经超过了截止时间，则此次遍历失败，进行回溯。

5.若当前情况下，该结点为村庄，更新参与者所带生活必需品数量。

6.若当前情况下，该结点为普通结点，检查该结点处生活必需品是否缺乏，若不缺乏，减去消耗的生活必需品数量，否则回溯到最近的村庄或起点，进行下一次遍历。

7.若当前情况下，该结点为矿山，考察挖矿以及不挖矿仅停留这两种情况，前一种情况使用类似第六步 的方法进行处理，后一种要加上挖矿的收益。

8.一次遍历后，回到第三步以输出结果或进行下一次遍历。



**图5.1.1 图回溯法流程图**

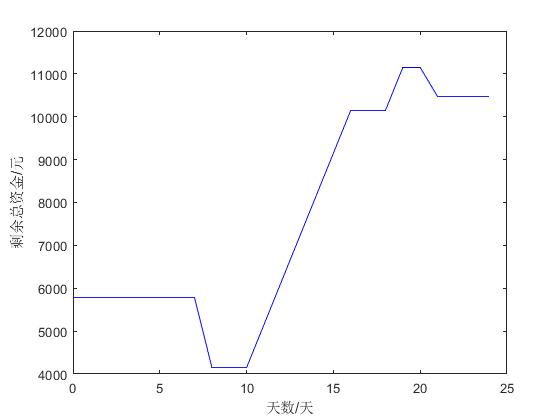
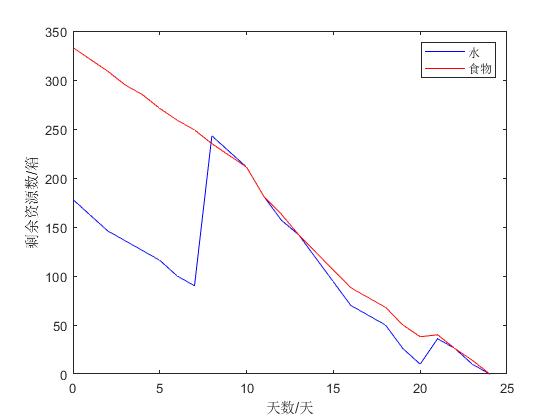
### 5.1.3 已知天气情况时一名玩家的最佳策略模型的结果分析

不挖矿情形下，玩家参与第一关的行走路线为：1→25→26→27，他需要在起点购买食物 38箱，水 42 箱，到达终点时剩余资金 9410 元;挖矿情况下，玩家参与第一关的行走路线如右图所示。从图中可以看出参与者从起点 1 出发，沿箭头走到 23 和9号结点后，由于沙暴天气需要停留一天;第一次到达村庄后玩家补给资源，接着经由 14 号结点到达矿山，在矿山停留 9 天，由于17、18 天是沙暴天气为避免损失不挖矿，其余 7 天挖矿;原路返回村庄再次补给资源后，经由9 号、21 号结点到达终点 27;整个过程历时 24 天，参与者在起点购买生活必需品数量及到达时资金剩余量如下表第一列所示。

**表5.1.2.2 挖矿时参与者在起点购买生活必需品数量及到达时资金剩余量**

|  |  |  |
| --- | --- | --- |
| 关卡编号 | 第一关 | 第二关 |
| 在起点购买食物/箱 | 333 | 405 |
| 在起点购买水/箱 | 178 | 130 |
| 到达终点资金剩余量/元 | 10470 | 12730 |

不挖矿与挖矿的情况相比，显然挖矿的情况得到的剩余资金数更大，因此我们采用这种策略作为最佳策略。根据图5.1.2 中所示的路线以及我们编写的程序，可以得到参与者每天的剩余资金数、剩余水量、剩余食物量，并将其填入Result.xlsx，同时借助 Matlab 绘制剩余资金数、资源剩余量的折线图，如下图所示。

**第一关剩余资金数变化图 第一关资源剩余量变化图**

1.剩余资金数的变化情况分析

如上左图所示：从第0天起资金保持在 5780 元不变，直到第 8天在村庄补给资源之后剩余资金开始减少，根据计算机结果玩家在到达矿山之后将挖矿 7 天，这一段时间资金剩余量持续上升，第 17、18 天是沙暴天气此时生活必需品的消耗量增加，为减少损失玩家选择在矿山原地停留两天，因此在第 15 天开始剩余资金数连续两天保持不变;离开矿山后玩家在前往终点的过程中需回村庄补给一次资金又一次减少，最终到达终点时剩余资金为10470 元。

2.资源剩余量的变化情况分析

如上右图所示：因为每千克水的价格为 1.67 元，每千克食物的价格为 5元，因此参与者选择在起点带尽可能多的食物，以免中途在村庄补给食物耗费更多金钱，玩家仅在第二次到达村庄时补给了少量食物，第一次到达村庄时仅补给了一部分水;食物和水的剩余量总的来说是不断减少的，且在到达终点时减少到0，所以该情况下参与者不会因为资源剩余而产生损失。

**第二关的结果分析**

对第二关的参与者挖矿和不挖矿经行比较，发现在第二关挖矿到达终点时的资金剩余量更多为12730元，因此我们采用挖矿策略作为最佳策略，玩家的行走路线如下图所示,从图中可以看出玩家在结点4、21、39因沙暴停留一天，两次经过矿山，挖矿天数分别为5天和8天，不会出因沙暴停止挖矿。

1

22

起点1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

7

28

29

31

32

33

34

35

36

37

38

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

56

57

58

59

60

61

63

终点

64

64

矿山

村庄

矿山

村庄

**图5.1.4 挖矿第二关行走路线**

第一，二关中玩家每天所在区城、剩余资金数、剩余水量、剩余食物量略表如下：

**表 1，2 关简略结果**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 第一关 | | | | | 第二关 | | | | |
| 日期 | 区域 | 剩余资金数/元 | 剩余水量/箱 | 剩余食物量/箱 | 日期 | 区域 | 剩余资金数/元 | 剩余水量/箱 | 剩余食物量/箱 |
| 0 | 1 | 5780 | 178 | 333 | 0 | 1 | 5300 | 130 | 405 |
| 1 | 25 | 5780 | 162 | 321 | 1 | 2 | 5300 | 114 | 393 |
| 2 | 24 | 5780 | 146 | 309 | 2 | 3 | 5300 | 98 | 381 |
| ...... | | | | | ...... | | | | |
| 23 | 21 | 10470 | 10 | 14 | 29 | 63 | 12730 | 16 | 12 |
| 24 | 27 | 10470 | 0 | 0 | 30 | 64 | 12730 | 0 | 0 |

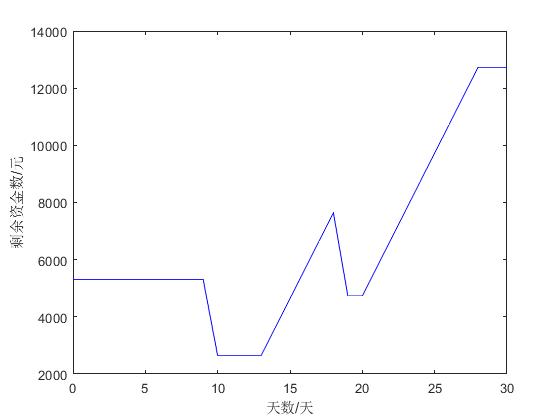
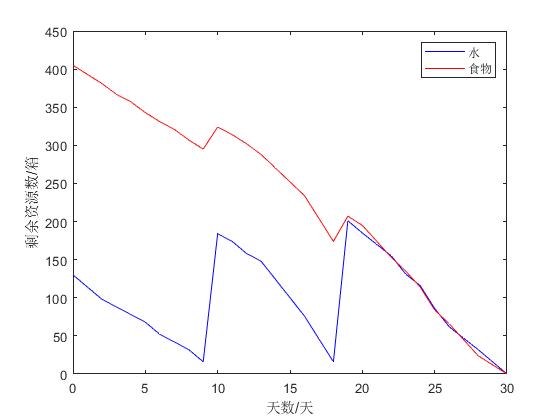
1.剩余资金数的变化情况分析

从图 5.1.5 左图可以看出，剩余资金量两次减少，两次增加，其减少是因为参与者经过村庄 39、62时均补给了一定量的资源，同时参与者两次前往矿山 55，第一次挖矿 5天，第二次挖矿8 天，都为其带来了一定的收入，因此剩余资金数两次增加。

2.资源剩余量的变化情况分析

与第一关的情况类似，这一关中，资源剩余量总体上也呈现减少的趋势，图像上的两次上升反映了参与者在村庄补给资源的情况;与第一关一样，这一关中参与者从起点出发时，购买的食物远多于水，这仍然是因为单位重量的食物的价格高于单位重量的水的价格，在起点带尽量多的食物可以尽可能的减少在村庄以两倍的价格补给食物造成的损失。

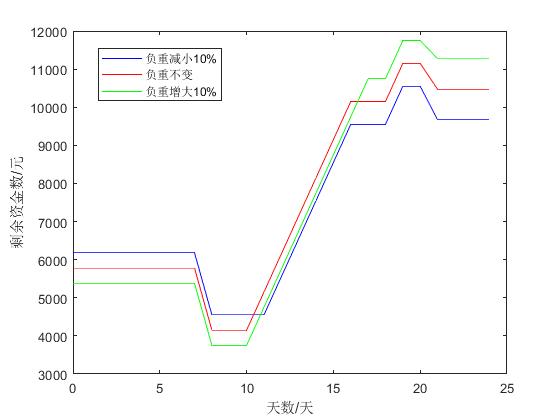
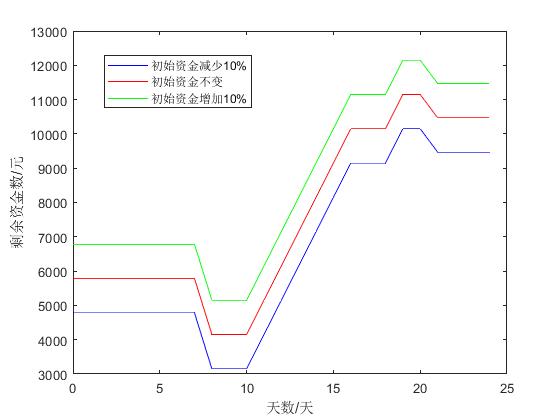
与第一关类似我们绘制剩余资金数、资源剩余量的折线图，如下图所示：

**第二关剩余资金数变化图 第二关资源剩余量变化图**

* **灵敏度分析**

为了判断最优模型对初始资金和负重上线的敏感性，因此我们针对挖矿情况下的模型进行灵敏度分析，我们将负重上限和初始资金前后变化10％，并作这几种况下剩余资金量的变化曲线，如下图所示。

**改变初始资金后剩余资金的变化图**  **改变初始资金后剩余资金的变化图**

左图中仅在到达终点之前的几天负重上限的改变对剩余资金量的影响略大;而右图中初始资金的改变一直对剩余资金量有较大影响，仅在参与者第一次在村庄补给资源时，初始资金的改变对剩余资金量的影响略小，可以很明显的看出其对参与者的决策无影响，所以曲线表现为上下平移;相较之下剩余资金量对初始资金的改变更加敏感。

## 5.2 问题二：仅知当天天气情况时一名玩家的最佳策略模型的建立与求解

由于只知道当天的天气情况，经过分析我们发现玩家决策与5种期望有很大的关系：每天行走所耗资源对应金额期望，挖矿净收益期望，挖矿消耗资源对应金额期望，挖矿所消耗食物和水的箱数期望，由于题一给出了30天的天气数据，我们可根据该数据确定每天每种天气出现的概率，同时结合第三关10天内无沙尘暴天气，和第四关的少有沙尘暴天气的情况，对概率经行适当的调整，结合天气概率计算出期望值，最后基于期望值建立了决策模型。

经过分析发现有以下几个决策点：

一般节点的决策：矿区，终点，村庄

到达村庄后的决策：购买多少箱水和食物

到达矿区后的决策：继续挖矿，停留，前往终点

### 5.2.1 仅知当天天气情况时一名玩家的最佳策略模型的建立

* 五种数学期望的计算：
* 每天行走所耗资源对应的金额的期望：

设每天行走所耗资源对应的金额的期望为 , 根据第三关、第四关的参数设定，晴 朗时水和食物的基础消耗量分别为 3 箱和 4 箱, 高温时水和食物的基础消耗量均为 9 箱， 根据游戏规则，行走时生活必需品的消耗量为基础消耗量的两倍，且沙暴天气下不能行 走，只能原地停留，因此



* 挖矿净收益的期望的计算：

计算挖矿净收益，我们需要考察两天的天气，设挖矿那一天的天气为 X ，从终点前一个结点到终点的那天的天气为 Y ，则 X 、 Y 均可能取 A 、 B 、 C ，因此二维随机变量 {X,Y} 共有 9 种取值组合; 我们假设沙漠中各天的天气相互独立，则有



我们记 当 时，挖矿净收益记为 , 当  时，挖矿净收益记为 , 白  时，挖矿净收益记为 , 则



该式中第一项  为参与者挖矿获得的基础收益，第二项为相比于正常行进，参与者在 挖矿所需多消耗的生活必需品对应的资金，第三项为参与者最后一天正常行进所消耗的 生活必需品对应的资金;  可用类似的方法进行计算，挖矿净收益的期望为:



* 挖矿所耗资源对应的金额的期望：

设挖矿所耗资源对应的金额的期望为 E3, 则可仿照 E1 的计算过程对其进行计算，即



从左到右分别为晴朗、高温、沙暴天气下挖矿所耗资源对应的金额的期望。

* 挖矿所耗水和食物箱数的期望：

设挖矿所耗水和食物箱数的期望分别为 E4 、 E5, 则使用与之前类似的方法可得



中括号里的内容为晴朗、高温、沙暴时基础消耗量的总和。

决策模型的构建

在构建决策模型，给出决策函数之前，我们先设 A,B 分别为矿山、村庄所在结点 的编号,  为整个过程中沙暴天数，  为参与者选择下一个结点的决策函数，其含

义为第 t天位于第 i 个结点的参与者应选择的下一个结点号，  为最多的挖矿天数, 其表达式为：



其中  为该关卡规定的到达终点的截止时间，d  为我们在 5.1.2 中定义的函数，表示从第 i 个结点到第j个结点的最短路径。

**不同决策下的条件限制**

* 直奔终点的情况：



情况一为挖矿的净收益为负数，情况二为挖矿的收益不足以弥补“绕路”产生的资金消耗，决策表明在这两种情况下玩家应选择最短路径直奔终点。

* 在一般节点选择前往矿区还是村庄

如果 , 使得



则参与者应选择先去村庄补给资源，再前往矿山; 该式中第一式的含义为：参与者在  时间内挖矿所得净收益超过了他 “绕路”前往村庄产生的资源消耗所对应的金额与他购 买的在  '时间内挖矿所需的资源所对应的金额的总和; 第二式的含义为: 参与者在村庄补给的资源重量没有超过他的负载上限，其中  、  分别为每箱水和食物的重量，  为参与者的载重上限，  为参与者从起点出发，到达村庄所用的时间，即



* 在矿区选择原地等待，挖矿还是前往终点**：**

参与者在矿山挖矿时，若时间不够或者所携带的生活必需品不够，即



则应选择直奔终点，决策函数为



设  为已知天气的在第  天挖矿所获净收益，若 , 则选择继续挖矿，若



即第二天挖矿所获净收益能够弥补参与者因等待而多消耗的生活必需品价格，则原地等待，因为后续继续挖矿仍可能获利，否则直奔终点，决策函数与式(5.2.2)相同。

根据以上条件以及决策函数设计算法即可得到玩家的最佳策略。

### 5.2.2 仅知当天天气情况时一名玩家的最佳策略模型的求解

通过上文的分析我们发现计算一些常量并将它们进行比较即可做出决策，因此我们的算法分为常量的初始化和迭代两部分进行，具体步骤如下：

1. 计算所需的各种常量，例如每种天气出现的概率、5 种数学期望以及水和食物的消耗量、任意两个结点之间的行走天数等。

2.计算实际挖矿时间  以及参与者的补给行为能支撑他多挖矿的天数  。

3. 根据上文所求的递推关系式以及各种期望值进行迭代，从而决定出发后直奔终点还是前往村庄或矿山，并输出剩余资金数、起点购买的水和食物的箱数。

由于题目中说明了第四关较少出现沙暴天气，为使结果更加准确我们对第一、二关的数据进行适当的调整：每 10 天中出现一次沙暴天气，又因为第三关中E2<0，所以参与者直奔终点，综上我们只需骤编写求解第四关的代码，同时对 E2 <0 时直奔终点的情况进行解释证明。

### 5.2.3 仅知当天天气情况时一名玩家的最佳策略模型的结果分析

由于沙漠中各天天气相互独立，且题目中提及10 天内不会出现沙暴天气，因此我们根据第一关、第二关的天气情况，计算这两天各种天气 组合出现的概率及各种天气情况下挖矿的净收益，如下表所示。

**表5.2.3 各种天气组合的概率以及各种天气下挖矿的净收益**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 天气组合 | X=A,Y=A | X=A,Y=B | X=B,Y=A | X=B,Y=B |
| 出现概率 | 9/64 | 15/64 | 15/64 | 25/64 |
| 挖矿净收益 | 35 | -125 | -45 | -205 |

由上表可得挖矿净收益的期望E2=-115<0，因为这两天均为晴天挖矿净收益最大，所以玩家应选择最短路径直奔终点，其路线为 1→5→6→13，但此时从起点前往终点需要5 天，而现行方案只需 3天，多走2天产生的费用为 220 元，挖矿产生的净收益仅为175元，在这种情况下是亏损的所以玩家应选择直奔终点。

**第四关的结果分析**

编写 C++代码求解第四关的决策方案，分析结果发现玩家大致的行走方向为∶ 起点→村庄→矿山→终点，经过的具体结点根据决策函数很容易可以得到。在这种行走方式下，到达终点时剩余资金数为 10065 元，参与者在起点购买的食品 200 箱，水 187 箱，这种方式为最优策略。

### 5.2.4 基于蒙特卡洛方法的模型合理性分析

**蒙特卡洛方法**

蒙特卡洛方法是一种基于随机数和统计抽样，以便近似求解数学物理问题的方法，这种方法又称统计实验法，或者计算机随机模拟方法，由冯·诺依曼命名;这种方法的典型应用有;求不规则图形的面积，求π的近似值等;在这里我们也可用蒙特卡洛方法对该问题中的模型进行仿真，以便检验该决策模型的合理性。

**对第四关使用蒙特卡洛方法进行仿真的结果分析**

我们使用蒙特卡洛方法对该模型进行仿真，即对于第四问的情况，使用 C++中的rand（）函数随机生成天气的情况，但此处各种天气的出现频率必须满足"较少出现沙暴"的要求，因此我们限定沙暴天气最多出现3 天，一次仿真后得到∶参与者到达终点时剩余资金数为 11245 元，参与者在起点购买的食品 190 箱，水 171 箱，可以发现该仿真结果与使用在第一关、第二关天气数据上稍作修改得到的结果类似，因此模型较为合理。

## 5.3 问题三模型的建立与求解

由于该问有多个玩家参与游戏, 并且玩家在游戏中的选择会受到其他玩家的影响，每个玩家为了实现自己的利益最大化必须考虑对方的行动策略。因此用博亦的模型来考虑，又因为各个玩家的决策不具有先后顺序所以采取静态博弈模型，第五关双方同时进行一次决策，为单阶段静态博亦。第六关双方进行多次决策为多阶段静态博亦。

### 5.3.1 问题三模型的建立

**1.博亦设定与求解目标**

我们首先考虑只有两名玩家的情况，假设两位玩家分别为 A，B且两人都是具有充分思维能力的理性人，能够充分考虑自己的策略和对方的选择。博弈的目标是使 A 能够在 B 按照符合 B 利益前提下行动时让自己获得最大的期望收益，因为A,B地位是平等的，策略集是完全对称的，因此我们设定的策略具有普适性。

经过分析我们发现可以采取的策略大概有两种：纯策略 [8]和混合策略，其中纯策略代表两个玩家使用同一种固定策略，走同条路线，混合策略代表玩家以从一个策略组  中以 的概率选择一种策略，给出的混合策略应当满足纳什均衡 [7]，即玩家选择任何一种策略最后的平均收益都是一样的。如果混合策略不满足纳什均衡，就存在好的策略和坏的 策略，玩家出于利己的角度，会倾向性地采用混合策略中的好策略，那么之前找到的混合策略就不能稳定存在了。

**2.博奕收益表的计算**

为了建立博亦收益表以对双方的策略进行分析，在接下来的部分中，我们把玩家从起点到终点消耗的水和食物的价格的相反数做为收益，并把失败失败的收益记为-10000(初始资金的相反数)。

策略组中几种策略的消耗量:

策略1. 第一天从 1 到 5, 第二天停留，第三到第四天从 5 走到终点,消耗的食物和水价值 465。

策略2.第一天到第三天沿  一直行走，第三天可到达终点，消耗的食

物和水的价格是 490。

策略3.沿着  的路线行走，并且只在晴天行走，消耗的食物和水的价格是 575 。

策略4.采用2的策略行走，但是假设对方采用2的行走策略，更改食物和水的购买方案。消耗的食物和水的价格是 980 。

策略5.采用2的策略行走，但是假设对方采用 1 的行走策略，更改食物和水的购买方案。消耗的食物和水的价格是 600。

策略6.采用1的策略行走，但是假设对方也采用该行走策略，更改食物和水的购买方案，消耗食物和水的价格为600。。

策略7.采用3的策略行走，但是假设对方也这么走，更改食物和水的购买方案,

消耗的价格是 1015。

**最优绝策略**

最优纯策略：

由于失败的代价非常大，保证存活且收益最大的方案就是最佳纯策略，对应为两名玩家都采用6 ，即到达终点时平均收益为-795且一定存活。

最优混合策略：

如果只采用 4和6两玩家采取策略的情况共有 4 种，对应的收益如下表所示：

|  |  |  |
| --- | --- | --- |
| 玩家A策略  玩家B策略 | 策略4 | 策略4 |
| 策略6 | (980,980) | (-800,-685) |
| 策略4 | (-685,-800) | (-795,-795) |

**表7 混合策略模型**

注释：（i，j）为玩家收益

如果两位玩家采用策略 4，6的概率分别为 P1,P2, 先寻找纳什平衡点让玩家采用 4和 6的收益相等：



求解得  ，即该策略组合不存在纳什均衡点，一般情况下玩家会严重倾向于使用策略6。再对组合1和3概率进行求解算得玩家到达终点时平均收益为-5260.2。失败率较大，在多人游戏中的平均收益也非常低,对策略6，7等两个策略的组合和部分 3 个策略的组合进行计算，发现在失败率等于 0 时不存在纳什均衡点，**综上在保证存活率的前提下采用纯策略较优。**

### 5.3.2模型的求解

通过上文的分析采用纯策略 6是两类方案中的最优方案。

图10 最优纯策略模型

终点13

起点1

2

3

4

5

6

7

8

10

11

12

* 允许交流情况下的最优策略

如果两玩家可以交流对策略1，3进行分配，在确保对两位玩家都是公平的条件下随机分配最优策略，则平均消耗的钱为

 ，相当于不交流的情况更优。

* 第六关三人多阶段博弈

第五关是单阶段博亦，采用混合博亦的模型解决，第六关是多阶段博亦玩家需要考虑长期利益，虽然不能直接交流，但可以通过观察分析对方之前的行为，达成某种程度上的合作，实现共赢，多阶段博亦的关键是寻找到纳什均衡点，因为纳什均衡点才能稳定存在，同时意味着每个玩家都有比较好的收益。根据 [6]中的命题10, 我们可以取消各个阶段的联系，寻找每个阶段的纳什均衡点 , 那么存在一个多阶段博亦的子博亦完全均衡，该完全均衡的路径与  相同。这个命题告诉我们，在这种游戏场景之下，如果对每一天进行孤立，在不考虑过去与未来的情况下寻找单日的纳什平衡点，那么把找到的 30 天的方案连起来看的话，就能构成 30 天方案中的一种平衡方案。利用该命题，可以把每一天看作一个阶段，分情况讨论，从而简化运算。

* **两玩家高温天抵达同一地点博弈**

如果两名玩家在同一高温天抵达了同一个地点，首先寻找纳什均衡点，基础消耗的价格是 135, 如果都行走每位玩家到达下一个点的消耗是 540 如果人停留一个人行走，那么停留的人到达下一个点的消耗是 405, 行走的人是 270 ，如果两人都停留那么消耗的期望必定大于 405, 因为最优情况下也只能做到做到一个人是405。表8展示出了以上分析结果，两人都停留的期望收益按照以下方式近似计算：停留之后，每人选择走或继续停留的概率是 1/2, 所以停留的消耗均为 585元。

|  |  |  |
| --- | --- | --- |
| 玩家A策略  玩家B策略 | 停留 | 行走 |
| 停留 | (-585，-585) | (-405,-270) |
| 行走 | (-270，405) | (-540,-540) |

**表8 行走策略分析**

注释：（i，j）为玩家收益

由上图可知玩家 A 行走的时候玩家B的最佳策略是停留; 当玩家 B 行走的时候，玩家 A 的最佳策略是行走，所以该点是一个纳什均衡点,由对称性可知 A 停留，B 行走也是一个纳什均衡点。由于玩家知道对方的食物和水的剩余情况，在互利的前提下可以进行一定的合作。在两者物资不等的情况下，物资多的玩家选择停留，物资少的玩家选择行走，当两者物资相等的时候，采用第五关中的混合博亦模型求解纳什均衡点，假设两位玩家选择停留和行走的概率分别为 , 有下列方程:



解得  。于是得到在双方物资相同时走到同一地点的策略：每位玩

家以 0.7 的概率选择行走，以 0.3 的概率选择停留。

* 两玩家高温天抵达同一座矿山博奕

如果两个玩家同时在矿山区，食物和水充足，那么容易得到表 9 所示的结果,分析下表可知无论对方采取什么策略，另外一方的最优策略都是挖矿，于是 A 挖矿B 挖矿构成纳什均衡点，说明在该情况下两玩家都应该挖矿。

|  |  |  |
| --- | --- | --- |
| 玩家A策略  玩家B策略 | 挖矿 | 停留 |
| 挖矿 | (95，95) | (595，-135) |
| 停留 | (-135，595) | (-135，-135) |

**表9 挖矿策略分析**

注释：（i，j）为玩家收益

* **两玩家高温天抵达村庄博弈**

由上文可知在未知天气的情况下玩家会购买更多的水和食物提升存活率，因此在抵达村庄的时候往往还留有一些水和食物。假设两人都准备购买基础价格为650的物资，然后前往矿山挖矿。选取从两人抵达村庄到两人均抵达矿山这段时间进行分析。容易得到，两人均购买然后行走的收益为-3140; 一人购买行走，一人停留的话，先离开村庄的人的收益为-650×2-270＋595 = -975，后离开的人的收益为-135-650×2-270=-1705。难点在于两人都停留的收益，这里近似认为如果都停留，那么之后两人都以50% 的概率选择购买离开或者继续停留。设每人的收益为 -135-x，那么有∶

1/4[(-135-x)+(-975)+(-1705) +(-3140] =-x

求得x = 1985，做出表10：

|  |  |  |
| --- | --- | --- |
| 玩家A策略  玩家B策略 | 停留 | 购买 |
| 停留 | (-2120，-2120) | (-1705，-975) |
| 购买 | (-975，-1705) | (-3140，-3140) |

**表10 村庄购买策略**

注释：（i，j）为玩家收益

不存在纯策略纳什均衡使用混合策略模型求得停留与购买的概率分别为55.6%，8和44.4%。

* 第三问一般策略

1.天气已知且存在多个玩家的单阶段博弈

由于中途失败造成的损失巨大，玩家的首要目的是生存、因此要在起点处购买足够多的食物和水，然后猜测其他玩家可能会采取的行走路线，这些路线是天气已知的单玩家模式下的较优行走策略，运用博弈论的方法寻找纳什均衡点，均衡点给出的策略就是玩家的行动策略。

2.天气未知且存在多个玩家的多阶段博弈

考虑到中途失败的损失巨大，玩家的首要目的依然是生存，因此要在起点处购买足够多的食物和水，按照天气未知单玩家场景下较优的行走路线行进，如果遇到与其他玩

### 5.3.3灵敏度分析

第三问的两人全局方案博弈和局部策略博弈都基于严格的博弈论基础，其结果对于各种情况也是普适的，不会因为输入的微小扰动而失效。

# 六、模型的评价与推广

## 6.1优点分析

1.整体模型充分地结合数学推导、算法实现、仿真模拟，对于问题给出了全面而深人的分析对于一个游戏策略问题，我们综合使用了图论、运筹学、统计学、博弈论等知识和工具进行分析，并熟练地编程实现。给出了对游戏"最优策略"的各种不同表述，如带有递推搜索思想的算法、统计的最优、决策的一般规则、博弈的平衡等，多角度地帮助玩家理解这个游戏并确定最优策略。

2.动态规划的算法实现进行充分的时间、空间复杂度优化

第一问的动态规划不仅给出全局最优解，我们还在分析算法时间、空间复杂度的基础上进行了彻底优化，达到单问题的秒级求解，使得后续的大量样本统计成为可能。

3.统计加随机模拟对于第三关随机天气下的策略设计给出了详细的论证

第三关的最终结果呈现后可能很容易猜到。但给出充分的思想来源和令人信服的论证并不容易。我们从统计结果抽取策略，并利用随机模拟较为完整地论证了该方案确实由于其他合理方案。

## 6.2缺点分析

1.利用确定天气情况下结果求解后两问时没有定量分析幸存者偏差，尽管我们可以通过动态规划回溯出优秀解，但在天气未知的情况下这些解有非常大的运气成分。最高收益和存活率二者是相互制衡的，而我们在分析一些优秀解的时候虽然也重点考虑了存活率，但无法显式地给出描述幸存者偏差的量并加以讨论。

2.给出的策略需要一定的算力支撑我们给出的有些策略难以通过直觉或人工计算快速得到验证，都需要一定的程序和算力进行实现，这些结果可能不易于被人从直观上理解。

3.对于多人玩家的情况没有给出完全最优解，虽然我们给出了局部最优策略，但对于三人的多阶段静态博弈没有给出完全最优解。由于博弈的过程难以由程序体现，最后的博弈过程没有进行模拟和全局计算。

## 6.3模型的改进

我们解决问题一中前往矿山挖矿过程的回溯算法，本质上是一种带有限界函数的深度优先遍历方法，为提高该算法的简明性，我们可以在现有代码之上，对其中的某些数据结构、搜索条件等进行优化，或采用粒子群算法等智能算法对这种情况进行求解;在问题二中，如果需要考虑参与者在村庄和矿山之间来回往返的情况，整个决策过程将变得更加复杂，很难在现有的过程上进行拓展，但可以设计与问题一中的回溯算法类似的算法来考察这种情况。

## 6.4模型的推广

我们建立模型的过程中用到的方法可以推广到其他领域，例如我们在求解问题一中使用到的贪心法就可以用于求解背包问题，该问题中涉及到的向背包中加入物品的过程与本问题中参与者在起点购买生活必需品的过程类似;在求解问题三时，我们参考了博弈论的相关理论，该理论可用于解决作战时兵力的分配、合作完成某项任务时人员的分配等多种实际问题。

# 七、参考文献

[1]周志华. 机器学习[M]. 北京∶清华大学出版社，2016.

[2]陈慧南. 算法设计与分析（C++语言描述）[M]. 北京∶ 电子工业出版社，2018.

[3]卓金武，王鸿钧.MATLAB 数学建模方法与实践（第三版）[M]. 北京∶北京航空航天大学出版社， 2018.

[4]林琦彤. 协同创新项目知识共享的演化博弈分析[D] 济南∶ 山东大学，2019.

[5]姜启源，谢金星，叶俊.数学模型（第四版）[M]. 北京∶高等教育出版社，2016.

[6]谢金星，薛毅.优化建模与LINDO/LINGO软件[M].北京∶清华大学出版社，2005.

[7]多阶段纳什均衡. http:// ａＣｕＩｔｙ ｂａａｓ．ｂｅｒｋｅＩｅｙ ｅｄｕｌｓｔａｄｅＩｉｓ／Ｇａｍｅ.

[8]strategy(game theory). htpp:// ｅｎ．ｗｉｋｉｐｅｄｉａ．ＯＩＢ／ｉｋｉｌｓｔｒａｔｅｓｙ

# 附 录

## 一、程序源代码

|  |
| --- |
| **问题1：求解第一关不挖矿时最佳策略的Matlab源代码** |
| p=xlsread('Result','Sheet2','B2:AB28');  for i=1:27  for j=i:27  p(j,i)=p(i,j);  end  end  clear i j;  inf=99999999;  n=size(p,1);  % 初始化距离矩阵  % 令无法直接到达的点之间的距离为inf  % 能直接到达的点之间距离为1  dis=p;  for i=1:27  for j=1:27  if dis(i,j)==0  dis(i,j)=inf;  end  end  end  % Floyd算法计算任意两点间的最短路径  for k=1:n  for i=1:n  for j=1:n  if dis(i,k)+dis(k,j)<dis(i,j)  dis(i,j)=dis(i,k)+dis(k,j);  end  end  end  end  dis=dis.\*(ones(27)-eye(27));  % 天气情况，1为晴朗，2为高温，3为沙暴  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];  w\_cost=[5 8 10]; % 三种情况下水的消耗量  f\_cost=[7 6 10]; % 三种情况下食物的消耗量  weight\_limit=1000; % 背包最大载重  day\_limit=30; % 截止日期  price\_w=5; % 水的价格  price\_f=10; % 食物的价格  weight\_w=3; % 水的重量  weight\_f=2; % 食物的重量  sumw=0; % 总共需要水的箱数  sumf=0; % 总共需要食物的箱数  dest=27; % 目的地编号  i=0;  day=0;  while day<dis(1,dest)  i=i+1;  if weather(i)~=3 % 没有遇到沙暴天气  day=day+1;  sumw=sumw+2\*w\_cost(weather(i));  sumf=sumf+2\*f\_cost(weather(i));  else % 遇到沙暴  sumw=sumw+w\_cost(weather(i));  sumf=sumf+f\_cost(weather(i));  end  end  left=10000-sumw\*price\_w-sumf\*price\_f;  disp("剩余资金："+left);  disp("所消耗水："+sumw);  disp("所消耗食物："+sumf); |

|  |
| --- |
| **问题1：求解第二关不挖矿时最佳策略的Matlab源代码** |
| p=zeros(64,64);  % 生成邻接矩阵  for i=1:64  a= ceil(i/8);  b= mod(i,8);  if b~=0  p(i,i+1)=1;  end  if b~=1  p(i,i-1)=1;  end  if mod(a,2)== 1  if a-1>0  if b==1  p(i,i+8)= 1;  p(i,i-8)= 1;  else  p(i,i+8)= 1;  p(i,i-8)= 1;  p(i,i+7)= 1;  p(i,i-9)= 1;  end  else  if b==1  p(i,i+8)= 1;  else  p(i,i+7)= 1;  p(i,i+8)= 1;  end  end  else  if a~=8  if b==0  p(i,i+8)=1;  p(i,i-8)=1;  else  p(i,i+8)=1;  p(i,i-8)=1;  p(i,i-7)=1;  p(i,i+9)=1;  end  else  if b==0  p(i,i-8)=1;  else  p(i,i-7)=1;  p(i,i-8)=1;  end  end  end  end  clear a b i;  inf=99999999;  n=size(p,1);  % 初始化路由矩阵  r=zeros(n,n);  for i=1:n  for j=1:n  r(i,j)=j;  end  end  % 初始化距离矩阵  dis=p;  for i=1:n  for j=1:n  if dis(i,j)==0  dis(i,j)=inf;  end  end  end  % floyd求最短路径  for k=1:n  for i=1:n  for j=1:n  if dis(i,k)+dis(k,j)<dis(i,j)  dis(i,j)=dis(i,k)+dis(k,j);  r(i,j)=r(i,k);  end  end  end  end  % 回推路径  arrow=zeros(1,n);  arrow(1)=1;  i=2;  pp=1;  while pp~=n  pp=r(pp,n);  arrow(i)=pp;  i=i+1;  end  dis=dis.\*(ones(n)-eye(n));  % 天气情况，1为晴朗，2为高温，3为沙暴  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];  w\_cost=[5 8 10]; % 三种情况下水的消耗量  f\_cost=[7 6 10]; % 三种情况下食物的消耗量  weight\_limit=1000; % 背包最大载重  day\_limit=30; % 截止日期  price\_w=5; % 水的价格  price\_f=10; % 食物的价格  weight\_w=3; % 水的重量  weight\_f=2; % 食物的重量  sumw=0; % 总共需要水的箱数  sumf=0; % 总共需要食物的箱数  dest=64; % 目的地编号  i=0;  day=0;  while day<dis(1,dest)  i=i+1;  if weather(i)~=3  day=day+1;  sumw=sumw+2\*w\_cost(weather(i));  sumf=sumf+2\*f\_cost(weather(i));  else  sumw=sumw+w\_cost(weather(i));  sumf=sumf+f\_cost(weather(i));  end  end  path="";  for i=1:n  if arrow(i)~=0  path=path+" "+arrow(i);  end  end  left=10000-sumw\*price\_w-sumf\*price\_f;  disp("最短路径长度："+dis(1,n));  disp("最短路径："+path);  disp("剩余资金："+left);  disp("所消耗水："+sumw);  disp("所消耗食物："+sumf); |
| **问题1：求解第一关挖矿时最佳策略的C++源代码** |
| #include<iostream>  #include<map>  using namespace std;  map<pair<int,int>,bool>mp;  const int qua[4]={0,1,2,3}; // 将图中的4个特殊点分为4类  // 按标号顺序记在该数组中  // 0起点，1村庄，2矿山，3终点  const int dist[4][4]={{0,6,8,3}, // 4个特殊点之间的距离矩阵  {6,0,2,3},  {8,2,0,5},  {3,3,5,0}};  const int f[4][4]={{0,1,1,1}, // 4类特殊点互相到达的决策情况  {0,0,1,1},  {0,1,0,1},  {0,0,0,0}};  const int wea[30]={2,2,1,3,1,2,3,1,2,2, // 天气情况,1,2,3分别代表晴朗、高温和沙暴  3,2,1,2,2,2,3,3,2,2,  1,1,2,1,3,2,1,1,2,2};  const int mx=3,my=2; // mx和my分别是水和食物的重量  const int cx=5,cy=10; // cx和cy分别是水和食物的基准价格  const int sx[4]={0,5,8,10};  //sx中下标为1-3的元素分别指晴朗,高温,沙暴天气下水的基础消耗  const int sy[4]={0,7,6,10};  //sy中下标为1-3的元素分别指晴朗,高温,沙暴天气下食物的基础消耗  const int n=4; // 共有4个特殊点  const int maxm=1200; // 背包容量  const int coins=10000; // 起始总资产  const int base=1000; // 挖矿每日收益  const int date=30; // 截止日期  int costx[32][4][4]; // 第d天从第i点走到第j点所消耗的水  int costy[32][4][4]; // 第d天从第i点走到第j点所消耗的食物  int days[32][4][4]; // 第d天从第i点走到第j点所需要的实际天数  int rec[32]; // 枚举不同情况下每天所在位置，会不断更新  // 每一天所到达的点的标记-1代表此时处于最短路径上的某个普通点或此时已经到达终点  // 其余的数字 分别代表当前玩家位于对应的特殊点 对应情况如qua数组所示  int act[32]; // 枚举不同情况下每天的行动，会不断更新  // 每一天的特殊行动情况 2代表挖矿 1代表于矿山停止行动 0代表在村庄购买  int ansx[32]; // ansx与ansact是最优解路径和最优解路径上的行为  int ansact[32];  int ansg,ansh; // ansg和ansh是最优解对应的初始水和食物的资源量  int ans=0; // ans用来存储剩余资金量，最后结果即本题答案  int g,h; // 用于枚举的初始水与食物资源量  // day代表日期，now代表当前位置，nm代表剩余重量，c代表剩余资金量，  // x代表水箱数，y代表食物箱数，type代表行动(-1代表常规操作)，type代表当天行动  void dfs(int day,int now,int nm,int c,int x,int y,int type)  {  rec[day]=now; // 更新位置  act[day]=type; // 更新状态  // 如果当前在终点  if(qua[now]==3)  {  if(ans<=c+x\*cx+y\*cy)  {  ansg=g;  ansh=h;  ans=c+x\*cx+y\*cy;  for(int i=0;i<=date;i++)  ansx[i]=rec[i];  for(int i=0;i<=date;i++)  ansact[i]=act[i];  }  act[day]=-1;  rec[day]=-1;  return;  }  // 如果超过截止日期  if(day>=date)  {  act[day]=-1;  rec[day]=-1;  return;  }  // 如果当前在村庄  if(qua[now]==1)  nm=maxm-mx\*x-my\*y; // 计算剩余重量  for(int i=0;i<n;i++)  if(f[qua[now]][qua[i]])  {  int tx=costx[day][now][i]; // 水的花销  int ty=costy[day][now][i]; // 食物的花销  int ucost=c; // c代表剩余资金  int ux,uy; // ux,uy代表剩余水和食物的数量  int um=nm; // nm代表剩余重量    if(x>=tx)  ux=x-tx;  else  {  ux=0;  ucost-=2\*(tx-x)\*cx;  um-=(tx-x)\*mx;  }    if(y>=ty)  uy=y-ty;  else  {  uy=0;  ucost-=2\*(ty-y)\*cy;  um-=(ty-y)\*my;  }    if(ucost<0||um<0)  continue;  dfs(day+days[day][now][i],i,um,ucost,ux,uy,0);  }  // 如果当前在矿山  if(qua[now]==2)  {  int attday=day; // 先存一下当前日期  int tx=sx[wea[attday]]; // 记录当天水的基本消耗  int ty=sy[wea[attday]]; // 记录当天食物的基本消耗  attday++;  // 判断水够不够当天基础消耗  if(x>=tx)  {  x-=tx;  tx=0;  }  else  {  tx-=x;  x=0;  }  // 判断食物够不够当天基础消耗  if(y>=ty)  {  y-=ty;  ty=0;  }  else  {  ty-=y;  y=0;  }  // 不够随时购买（任意处都是村庄）  nm-=tx\*mx+ty\*my;  c-=2\*tx\*cx+2\*ty\*cy;  if(nm>=0&&c>=0)  dfs(attday,now,nm,c,x,y,1);    attday=day;  tx=sx[wea[attday]]\*2; // 记录当天挖矿水的消耗  ty=sy[wea[attday]]\*2; // 记录当天挖矿食物的消耗  attday++;  // 判断水够不够当天挖矿消耗  if(x>=tx)  {  x-=tx;  tx=0;  }  else  {  tx-=x;  x=0;  }  // 判断食物够不够当天挖矿消耗  if(y>=ty)  {  y-=ty;  ty=0;  }  else  {  ty-=y;  y=0;  }  // 不够随时购买（任意处都是村庄）  nm-=tx\*mx+ty\*my;  c-=2\*tx\*cx+2\*ty\*cy;  c+=base;  if(nm>=0&&c>=0)  dfs(attday,now,nm,c,x,y,2);  }  // 回溯  rec[day]=-1;  act[day]=-1;  }  int main()  { // 初始化位置数组和行动数组  for(int d=0;d<=date;d++)  {  rec[d]=-1;  act[d]=-1;  }  // 初始化costx,costy,days矩阵，只有四个点  for(int d=0;d<date;d++)  for(int i=0;i<n;i++)  for(int j=0;j<n;j++)  if(f[qua[i]][qua[j]]) // f是决策矩阵  { // sumx代表消耗水的数量，sumy代表消耗食物的数量  int now=0,count=0,sumx=0,sumy=0;  while(count<dist[i][j])  {  if(wea[now+d]!=3) // 不为沙暴  {  count++; // count代表行走距离  sumx+=2\*sx[wea[now+d]];  sumy+=2\*sy[wea[now+d]];  }  else  {  sumx+=sx[wea[now+d]];  sumy+=sy[wea[now+d]];  }  now++; // now代表消耗时间  if(now+d>=date) // d代表当前日期  break;  }  // 第d天从i走不到j的情况  if(count<dist[i][j])  {  sumx=sumy=20000; // 随便赋值，超过金额最大量就行  now=30; // 随便赋值，超过29就行  }  costx[d][i][j]=sumx; // 消耗食物量  costy[d][i][j]=sumy; // 消耗水量  days[d][i][j]=now; // 消耗天数  }  // 枚举水占的重量  for(int i=0;i<=maxm;i+=mx)  { // g,h代表初始水和食物的数量  g=i/mx; // 水的数量，向下取整  h=(maxm-i)/my; // 食物数量，向下取整  if(!mp[make\_pair(g,h)]) // 如果不曾递归过这种情况，就进行递归  dfs(0,0,0,coins-g\*cx-h\*cy,g,h,-1);  mp[make\_pair(g,h)]=true; // 遍历过就改为true  }  // ansx代表所处位置，ansact代表当前行动  for(int i=0;i<=date;i++)  cout<<i<<":"<<ansx[i]<<";"<<ansact[i]<<endl;  cout<<endl;  cout<<ans<<" "<<ansg<<" "<<ansh<<endl;  } |

|  |
| --- |
| **问题1：求解第二关挖矿时最佳策略的C++源代码** |
| #include<iostream>  #include<map>  using namespace std;  map<pair<int,int>,bool>mp;  const int qua[6]={0,2,1,2,1,3}; // 将图中的6个特殊点分为4类  // 按标号顺序记在该数组中  // 0起点，1村庄，2矿山，3终点  const int dist[6][6]={{0,7,8,9,9,11}, // 6个特殊点之间的距离矩阵  {7,0,1,3,4,4},  {8,1,0,2,3,3},  {9,3,2,0,1,2},  {9,4,3,1,0,2},  {11,4,3,2,2,0}};  const int f[4][4]={{0,1,1,1}, // 6类特殊点互相到达的决策情况  {0,0,1,1},  {0,1,0,1},  {0,0,0,0}};  const int wea[30]={2,2,1,3,1,2,3,1,2,2, // 天气情况,1,2,3分别代表晴朗、高温和沙暴  3,2,1,2,2,2,3,3,2,2,  1,1,2,1,3,2,1,1,2,2};  const int mx=3,my=2; // mx和my分别是水和食物的重量  const int cx=5,cy=10; // cx和cy分别是水和食物的基准价格  const int sx[4]={0,5,8,10};  //sx中下标为1-3的元素分别指晴朗,高温,沙暴天气下水的基础消耗  const int sy[4]={0,7,6,10};  //sy中下标为1-3的元素分别指晴朗,高温,沙暴天气下食物的基础消耗  const int n=6; // 共有6个特殊点  const int maxm=1200; // 背包容量  const int coins=10000; // 起始总资产  const int base=1000; // 挖矿每日收益  const int date=30; // 截止日期  int costx[32][6][6]; // 第d天从第i点走到第j点所消耗的水  int costy[32][6][6]; // 第d天从第i点走到第j点所消耗的食物  int days[32][6][6]; // 第d天从第i点走到第j点所需要的实际天数  int ans=0;  int rec[32];  // 每一天所到达的点的标记-1代表此时处于最短路径上的某个普通点或此时已经到达终点  // 其余的数字 分别代表当前玩家位于对应的特殊点 对应情况如qua数组所示  int act[32];  // 每一天的特殊行动情况 2代表挖矿 1代表于矿山停止行动 0代表在村庄购买  int ansx[32]; // ansx与ansact是最优解路径和最优解路径上的行为  int ansact[32];  int ansg,ansh; // ansg和ansh是最优解对应的初始水和食物的资源量  int g,h; // 用于枚举的初始水与食物资源量  void dfs(int day,int now,int nm,int c,int x,int y,int type)  {  act[day]=type;  rec[day]=now;    if(qua[now]==3)  {  if(ans<=c+x\*cx+y\*cy)  {  ansg=g;  ansh=h;  ans=c+x\*cx+y\*cy;  for(int i=0;i<=date;i++)  ansx[i]=rec[i];  for(int i=0;i<=date;i++)  ansact[i]=act[i];  }  act[day]=-1;  rec[day]=-1;  return;  }  if(day>=date)  {  act[day]=-1;  rec[day]=-1;  return;  }  if(qua[now]==1)  nm=maxm-mx\*x-my\*y;  for(int i=0;i<n;i++)  if(f[qua[now]][qua[i]])  {  int tx=costx[day][now][i];  int ty=costy[day][now][i];  int ucost=c;  int ux,uy;  int um=nm;  if(x>=tx)  ux=x-tx;  else  {  ux=0;  ucost-=2\*(tx-x)\*cx;  um-=(tx-x)\*mx;  }    if(y>=ty)  uy=y-ty;  else  {  uy=0;  ucost-=2\*(ty-y)\*cy;  um-=(ty-y)\*my;  }  if(ucost<0||um<0)  continue;  dfs(day+days[day][now][i],i,um,ucost,ux,uy,0);  }  if(qua[now]==2)  {  int attday=day;  int tx=sx[wea[attday]];  int ty=sy[wea[attday]];  attday++;  if(x>=tx)  {  x-=tx;  tx=0;  }  else  {  tx-=x;  x=0;  }    if(y>=ty)  {  y-=ty;  ty=0;  }  else  {  ty-=y;  y=0;  }  nm-=tx\*mx+ty\*my;  c-=2\*tx\*cx+2\*ty\*cy;  if(nm>=0&&c>=0)  dfs(attday,now,nm,c,x,y,1);    attday=day;  tx=sx[wea[attday]]\*2;  ty=sy[wea[attday]]\*2;  attday++;  if(x>=tx)  {  x-=tx;  tx=0;  }  else  {  tx-=x;  x=0;  }    if(y>=ty)  {  y-=ty;  ty=0;  }  else  {  ty-=y;  y=0;  }  nm-=tx\*mx+ty\*my;  c-=2\*tx\*cx+2\*ty\*cy;  c+=base;  if(nm>=0&&c>=0)  dfs(attday,now,nm,c,x,y,2);  }  rec[day]=-1;  act[day]=-1;  }  int main()  {  for(int d=0;d<=date;d++)  {  rec[d]=-1;  act[d]=-1;  }  for(int d=0;d<date;d++)  for(int i=0;i<n;i++)  for(int j=0;j<n;j++)  if(f[qua[i]][qua[j]])  {  int now=0,count=0,sumx=0,sumy=0;  while(count<dist[i][j])  {  if(wea[now+d]!=3)  {  count++;  sumx+=2\*sx[wea[now+d]];  sumy+=2\*sy[wea[now+d]];  }  else  {  sumx+=sx[wea[now+d]];  sumy+=sy[wea[now+d]];  }  now++;  if(now+d>=date)  break;  }  if(count<dis[i][j])  {  sumx=sumy=20000;  now=30;  }  costx[d][i][j]=sumx;  costy[d][i][j]=sumy;  days[d][i][j]=now;  }  for(int i=0;i<=maxm;i+=mx)  {  g=i/mx;  h=(maxm-i)/my;  if(!mp[make\_pair(g,h)])  dfs(0,0,0,coins-g\*cx-h\*cy,g,h,-1);  mp[make\_pair(g,h)]=1;  }  for(int i=0;i<=date;i++)  cout<<i<<":"<<ansx[i]<<";"<<ansact[i]<<endl;  cout<<endl;  cout<<ans<<" "<<ansg<<" "<<ansh<<endl;  } |

|  |
| --- |
| **问题1：绘制第一关、第二关剩余资金数变化曲线的Matlab源代码** |
| t=[0:24];  money=xlsread('Result.xlsx','sheet1','C4:C28');  plot(t,money,'b') %第一关剩余资金  %% 绘制第二关挖矿时剩余资金变化情况的Matlab源代码  t=[0:30];  money=xlsread('Result.xlsx','sheet1','I4:I34');  plot(t,money,'b') %第二关剩余资金 |

|  |
| --- |
| **问题1：绘制第一关、第二关剩余水量、食物量变化曲线的Matlab源代码** |
| %% 第一关  t=[0:24];  water=xlsread('Result.xlsx','sheet1','D4:D28');  food=xlsread('Result.xlsx','sheet1','E4:E28');  plot(t,water,'b');  hold on;  plot(t,food,'r');  legend('水','食物') %第一关剩余资源  %% 第二关  t=[0:30];  water=xlsread('Result.xlsx','sheet1','J4:J34');  food=xlsread('Result.xlsx','sheet1','K4:K34');  plot(t,water,'b');  hold on;  plot(t,food,'r');  legend('水','食物') %第二关剩余资源 |

|  |
| --- |
| **问题1：以第一关为例进行灵敏度分析的Matlab源代码** |
| %% 改变负重上限：  t=[0:24];  money1=xlsread('sensitivity.xlsx','sheet1','A2:A26');  plot(t,money1,'b'); %负重减少10%的剩余资金数  hold on;  money2=xlsread('sensitivity.xlsx','sheet1','B2:B26');  plot(t,money2,'r'); %负重不变的剩余资金数  hold on;  money3=xlsread('sensitivity.xlsx','sheet1','C2:C26');  plot(t,money3,'g'); %负重增加10%的剩余资金数  legend('负重减小10%','负重不变','负重增大10%')  %% 改变初始资金：  t=[0:24];  money1=xlsread('sensitivity.xlsx','sheet1','E2:E26');  plot(t,money1,'b'); %初始资金减少10%的剩余资金数  hold on;  money2=xlsread('sensitivity.xlsx','sheet1','F2:F26');  plot(t,money2,'r'); %初始资金不变的剩余资金数  hold on;  money3=xlsread('sensitivity.xlsx','sheet1','G2:G26');  plot(t,money3,'g'); %初始资金增加10%的剩余资金数  legend('初始资金减少10%','初始资金不变','初始资金增加10%') |

|  |
| --- |
| **问题2：求解第四关最佳策略并使用蒙特卡洛方法进行仿真的C++源代码** |
| #include<iostream>  #include<map>  #include<time.h>  #include<stdlib.h>  using namespace std;  map<pair<int,int>,bool>mp;  const int qua[4]={0,1,2,3};  const int dist[4][4]={{0,5,5,8},  {5,0,2,3},  {5,2,0,3},  {8,3,3,0}};  const int f[4][4]={{0,1,1,1},  {0,0,1,1},  {0,1,0,1},  {0,0,0,0}};  int wea[30]={2,2,1,3,1,2,2,1,2,2,  1,2,1,2,2,2,3,1,2,2,  1,1,2,1,3,2,1,1,2,2};  const int dis[26][26]={  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,1,2,3,4,1,2,3,4,5,2,3,4,5,6,3,4,5,6,7,4,5,6,7,8,  0,1,0,1,2,3,2,1,2,3,4,3,2,3,4,5,4,3,4,5,6,5,4,5,6,7,  0,2,1,0,1,2,3,2,1,2,3,4,3,2,3,4,5,4,3,4,5,6,5,4,5,6,  0,3,2,1,0,1,4,3,2,1,2,5,4,3,2,3,6,5,4,3,4,7,6,5,4,5,  0,4,3,2,1,0,5,4,3,2,1,6,5,4,3,2,7,6,5,4,3,8,7,6,5,4,  0,1,2,3,4,5,0,1,2,3,4,1,2,3,4,5,2,3,4,5,6,3,4,5,6,7,  0,2,1,2,3,4,1,0,1,2,3,2,1,2,3,4,3,2,3,4,5,4,3,4,5,6,  0,3,2,1,2,3,2,1,0,1,2,3,2,1,2,3,4,3,2,3,4,5,4,3,4,5,  0,4,3,2,1,2,3,2,1,0,1,4,3,2,1,2,5,4,3,2,3,6,5,4,3,4,  0,5,4,3,2,1,4,3,2,1,0,5,4,3,2,1,6,5,4,3,2,7,6,5,4,3,  0,2,3,4,5,6,1,2,3,4,5,0,1,2,3,4,1,2,3,4,5,2,3,4,5,6,  0,3,2,3,4,5,2,1,2,3,4,1,0,1,2,3,2,1,2,3,4,3,2,3,4,5,  0,4,3,2,3,4,3,2,1,2,3,2,1,0,1,2,3,2,1,2,3,4,3,2,3,4,  0,5,4,3,2,3,4,3,2,1,2,3,2,1,0,1,4,3,2,1,2,5,4,3,2,3,  0,6,5,4,3,2,5,4,3,2,1,4,3,2,1,0,5,4,3,2,1,6,5,4,3,2,  0,3,4,5,6,7,2,3,4,5,6,1,2,3,4,5,0,1,2,3,4,1,2,3,4,5,  0,4,3,4,5,6,3,2,3,4,5,2,1,2,3,4,1,0,1,2,3,2,1,2,3,4,  0,5,4,3,4,5,4,3,2,3,4,3,2,1,2,3,2,1,0,1,2,3,2,1,2,3,  0,6,5,4,3,4,5,4,3,2,3,4,3,2,1,2,3,2,1,0,1,4,3,2,1,2,  0,7,6,5,4,3,6,5,4,3,2,5,4,3,2,1,4,3,2,1,0,5,4,3,2,1,  0,4,5,6,7,8,3,4,5,6,7,2,3,4,5,6,1,2,3,4,5,0,1,2,3,4,  0,5,4,5,6,7,4,3,4,5,6,3,2,3,4,5,2,1,2,3,4,1,0,1,2,3,  0,6,5,4,5,6,5,4,3,4,5,4,3,2,3,4,3,2,1,2,3,2,1,0,1,2,  0,7,6,5,4,5,6,5,4,3,4,5,4,3,2,3,4,3,2,1,2,3,2,1,0,1,  0,8,7,6,5,4,7,6,5,4,3,6,5,4,3,2,5,4,3,2,1,4,3,2,1,0  };  const int quai[4]={1,14,18,25};  const int mx=3,my=2;  const int cx=5,cy=10;  const int sx[4]={0,3,9,10};  const int sy[4]={0,4,9,10};  const int n=4; // 共有4个特殊点  const int maxm=1200; // 背包容量  const int coins=10000; // 起始总资产  const int base=1000; // 挖矿每日收益  const int date=30; // 截止日期  int costx[32][4][4]; // 第d天从第i点走到第j点所消耗的水  int costy[32][4][4]; // 第d天从第i点走到第j点所消耗的食物  int days[32][4][4]; // 第d天从第i点走到第j点所需要的实际天数  int ans=0; // ans用来存储剩余资金量，最后结果即本题答案  int rec[32]; // 枚举不同情况下每天所在位置，会不断更新  // 每一天所到达的点的标记-1代表此时处于最短路径上的某个普通点或此时已经到达终点  // 其余的数字 分别代表当前玩家位于对应的特殊点 对应情况如qua数组所示  int act[32]; // 枚举不同情况下每天的行动，会不断更新  // 每一天的特殊行动情况 2代表挖矿 1代表于矿山停止行动 0代表在村庄购买  int ansx[32]; // ansx与ansact是最优解路径和最优解路径上的行为  int ansact[32];  int ansg,ansh; // ansg和ansh是最优解对应的初始水和食物的资源量  int g,h;  double e[4]={0};  double p[4]={0,17.0/30,1.0/3,1.0/10};  double pi[10];  int main()  {  // srand((unsigned)time(NULL));  // int count3=0;  // for(int i=1;i<=30;i++){  // if(count3==3){  // wea[i]=rand()%2+1;  // }else{  // wea[i]=rand()%3+1;  // if(wea[i]==3){  // count3++;  // }  // }  // }    for(int i=1;i<=30;i++){  cout<<wea[i]<<" ";  }      for(int i=1;i<=2;i++){  e[1]=e[1]+p[i]\*(2\*sx[i]\*cx+2\*sy[i]\*cy);  }    for(int i=1;i<=3;i++){  e[3]=e[3]+p[i]\*(2\*sx[i]\*cx+2\*sy[i]\*cy);  }    double ew=3\*3\*p[1]+3\*9\*p[2]+3\*10\*p[3],ef=3\*4\*p[1]+3\*9\*p[2]+3\*10\*p[3];  for(int i=1;i<=3;i++){  for(int j=1;j<=3;j++){  if(j!=3){  pi[(i-1)\*3+j]=(base-sx[i]\*cx-sy[i]\*cy-2\*sx[j]\*cx-2\*sy[j]\*cy)\*p[i]\*p[j];  }else{  pi[(i-1)\*3+j]=(base-sx[i]\*cx-sy[i]\*cy-(2\*3\*cx+2\*4\*cy)\*p[1]-(2\*9\*cx+2\*9\*cy)\*p[2])\*p[i]\*p[j];  }  }  }  for(int i=1;i<=9;i++){  e[2]=e[2]+pi[i];  }    int tmax=30,nd=3,T0=tmax-dis[1][quai[2]]-dis[quai[2]][n]-nd;  int bw=178,bf=333;  for(int d=0;d<date;d++)  for(int i=0;i<n;i++)  for(int j=0;j<n;j++)  if(f[qua[i]][qua[j]]){  int now=0,count=0,sumx=0,sumy=0;  while(count<dist[i][j]){  if(wea[now+d]!=3){  count++;  sumx+=2\*sx[wea[now+d]];  sumy+=2\*sy[wea[now+d]];  }else{  sumx+=sx[wea[now+d]];  sumy+=sy[wea[now+d]];  }  now++;  if(now+d>=date)  break;  }  if(count<dist[i][j]){  sumx=sumy=20000;  now=30;  }  costx[d][i][j]=sumx;  costy[d][i][j]=sumy;  days[d][i][j]=now;  }  double T0\_1=min((bw-costx[0][0][1]-(dis[quai[1]-1][quai[2]]+dis[quai[2]][25])\*(2\*3\*p[1]+2\*9\*p[2]))/ew,  (bf-costy[0][0][1]-(dis[quai[1]-1][quai[2]]+dis[quai[2]][25])\*(2\*4\*p[1]+2\*9\*p[2]))/ef);    int i=1,t\_m=0;  double maxx=-99999999;  while(i<=T0-T0\_1){  if(i\*e[2]>e[1]\*(dist[0][1]+dist[1][2]-dist[0][2])+2\*i\*e[3]&&mx\*(bw-costx[0][0][1]+  i\*ew)+my\*(bf-costy[0][0][1]+i\*ef)<=maxm){  if(i\*e[2]-e[1]\*(dist[0][1]+dist[1][2]-dist[0][2])-2\*i\*e[3]>maxx){  maxx=i\*e[2]-e[1]\*(dist[0][1]+dist[1][2]-dist[0][2])-2\*i\*e[3];  t\_m=i;  }  }  i=i+1;  }    int dayss=0;  int nc=coins;  nc=nc-cx\*bw-cy\*bf;  cout<<nc<<"\n";  bw=bw-costx[0][0][1];  bf=bf-costy[0][0][1];  dayss=dayss+days[0][0][1];    bw=bw+t\_m\*ew;  bf=bf+t\_m\*ef;  cout<<t\_m\*ew<<" "<<t\_m\*ef<<endl;    nc=nc-t\_m\*ew\*2\*cx-t\_m\*ef\*2\*cy;    bw=bw-costx[dayss][1][2];  bf=bf-costy[dayss][1][2];  dayss=dayss+days[dayss][1][2];  int k=0;  for(;;k++){  bw=bw-3\*sx[wea[dayss+k]];  bf=bf-3\*sy[wea[dayss+k]];    if(date-k<=days[dayss+k][2][3]||bw<=2\*9\*dist[2][3]||bf<=2\*9\*dist[2][3]){  bw=bw+3\*sx[wea[dayss+k]];  bf=bf+3\*sy[wea[dayss+k]];  break;  }  }  k=k-1;  nc=nc+base\*k;  cout<<bw<<" "<<bf<<endl;  cout<<k<<" "<<nc<<" "<<endl;  } |

|  |
| --- |
| **问题3：求解决策方案的Lingo源代码** |
|  |

|  |
| --- |
| **问题3：求解第六关最佳策略的C++源代码** |
|  |

## 二、详细表格

**附表2.1：第1、2关的相应结果（即Result.xlsx）**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **第一关** | | | | | **第二关** | | | | |
| **日期** | **所在**  **区域** | **剩余资金数/元** | **剩余水量/箱** | **剩余食物量/箱** | **日期** | **所在**  **区域** | **剩余资金数/元** | **剩余水量/箱** | **剩余食物量/箱** |
| 0 | 1 | 5780 | 178 | 333 | 0 | 1 | 5300 | 130 | 405 |
| 1 | 25 | 5780 | 162 | 321 | 1 | 2 | 5300 | 114 | 393 |
| 2 | 24 | 5780 | 146 | 309 | 2 | 3 | 5300 | 98 | 381 |
| 3 | 23 | 5780 | 136 | 295 | 3 | 4 | 5300 | 88 | 367 |
| 4 | 23 | 5780 | 126 | 285 | 4 | 4 | 5300 | 78 | 357 |
| 5 | 21 | 5780 | 116 | 271 | 5 | 12 | 5300 | 68 | 343 |
| 6 | 9 | 5780 | 100 | 259 | 6 | 21 | 5300 | 52 | 331 |
| 7 | 9 | 5780 | 90 | 249 | 7 | 21 | 5300 | 42 | 321 |
| 8 | 15 | 4150 | 243 | 235 | 8 | 29 | 5300 | 32 | 307 |
| 9 | 13 | 4150 | 227 | 223 | 9 | 38 | 5300 | 16 | 295 |
| 10 | 12 | 4150 | 211 | 211 | 10 | 39 | 2640 | 184 | 324 |
| 11 | 12 | 5150 | 181 | 181 | 11 | 39 | 2640 | 174 | 314 |
| 12 | 12 | 6150 | 157 | 163 | 12 | 46 | 2640 | 158 | 302 |
| 13 | 12 | 7150 | 142 | 142 | 13 | 55 | 2640 | 148 | 288 |
| 14 | 12 | 8150 | 118 | 124 | 14 | 55 | 3640 | 124 | 270 |
| 15 | 12 | 9150 | 94 | 106 | 15 | 55 | 4640 | 100 | 252 |
| 16 | 12 | 10150 | 70 | 88 | 16 | 55 | 5640 | 76 | 234 |
| 17 | 12 | 10150 | 60 | 78 | 17 | 55 | 6640 | 46 | 204 |
| 18 | 12 | 10150 | 50 | 68 | 18 | 55 | 7640 | 16 | 174 |
| 19 | 12 | 11150 | 26 | 50 | 19 | 62 | 4730 | 201 | 207 |
| 20 | 13 | 11150 | 10 | 38 | 20 | 55 | 4730 | 185 | 195 |
| 21 | 15 | 10470 | 36 | 40 | 21 | 55 | 5730 | 170 | 174 |
| 22 | 9 | 10470 | 26 | 26 | 22 | 55 | 6730 | 155 | 153 |
| 23 | 21 | 10470 | 10 | 14 | 23 | 55 | 7730 | 131 | 135 |
| 24 | 27 | 10470 | 0 | 0 | 24 | 55 | 8730 | 116 | 114 |
| 25 |  |  |  |  | 25 | 55 | 9730 | 86 | 84 |
| 26 |  |  |  |  | 26 | 55 | 10730 | 62 | 66 |
| 27 |  |  |  |  | 27 | 55 | 11730 | 47 | 45 |
| 28 |  |  |  |  | 28 | 55 | 12730 | 32 | 24 |
| 29 |  |  |  |  | 29 | 63 | 12730 | 16 | 12 |
| 30 |  |  |  |  | 30 | 64 | 12730 | 0 | 0 |

## 三、支撑材料内容组成

|  |  |  |
| --- | --- | --- |
| 文件夹 | 文件名 | 主要功能/用途 |
| 源代码 | p1\_1d.m | 求解第一关不挖矿时的最佳策略（Matlab源代码） |
| p2\_2d.m | 求解第二关不挖矿时的最佳策略（Matlab源代码） |
| p1\_1.cpp | 求解第一关挖矿时的最佳策略（C++源代码） |
| p1\_2.cpp | 求解第二关挖矿时的最佳策略（C++源代码） |
| plot1\_m | 绘制第一关、第二关剩余资金数的变化曲线（Matlab源代码） |
| plot2\_m | 绘制第一关、第二关剩余水量、食物量的变化曲线  （Matlab源代码） |
| sensitivity1.m | 以第一关为例进行灵敏度分析（Matlab源代码） |
| p2\_4.cpp | 求解第四关的最佳策略，并使用蒙特卡洛方法进行仿真  （C++源代码） |
| 数据 | sensitivity1.xlsx | 该表格是我们以第一关为例进行灵敏度分析时的原始数据表格，该表格用于导入Matlab进行计算 |
| Result.xlsx | 该表格是我们第一问中第一关与第二关分别求出的最优解对应的每天的水、食物和资金的剩余量，及第一关地图的邻接矩阵，运行p1\_1d.m文件时需要导入该表格 |