# 穿越沙漠游戏中的玩家决策分析

## 摘要

在穿越沙漠的游戏中，游戏目标为在规定时间内到达终点，并保留尽可能多的资金。玩家在起点处使用初始资金购买不超过负重上限的水和食物，依据地图在天气情况变化的沙漠中穿行，途中可在矿山挖矿补充资金或在村庄购买物资。为给出玩家游戏过程中的一般最佳策略，需要结合不同的游戏设定条件进行分析。

针对问题一，求解单一玩家在全过程天气情况完全已知时的一般最优游戏策略，我们使用弗洛伊德算法得到两关键点之间的最短路径，随后考虑直奔终点不挖矿和尽量多挖矿两条游戏思路，分别使用最短路径法和**回溯算法**求解，最后得到此种情景下第一关最大资金剩余量为 10470 元，第二关最大资金剩余量为 12730 元。

针对问题二，求解单一玩家在仅知当天天气情况决定当天行动时的一般最优游戏策略，我们利用问题一中已知的天气数据，结合第三关和第四关的题设天气情况进行调整，使用**马尔可夫链**预测天气情况。第三关由于不存在沙暴天气，仅仅使用期望求资金剩余量和弗洛伊德算法求最短路路径即可得出答案。第四关利用**蒙特卡罗算法**模拟天气，然后利用问题一中的算法对实际情况进行了 100 次模拟，作为玩家在开局时的决策依据，决定购买水和食物的比例以及大致路线。由于玩家需要根据天气变化动态制定计划，我们选取了地图中的关键点，利用期望计算出了各个关键点决策的标准，将以上两种策略相结合作为玩家的参考依据。**结果表明第三关无论天气如何玩家都应选择直奔终点，第四关玩家将选择去挖矿，但补给次数、挖矿次数会根据天气情况来进行具体决策**。最后，从模拟的 100 次结果中我们可以发现有几条较优的路径，我们根据玩家对于**风险的接受程度**对其进行分类，为不同风险接受程度的玩家提供了游戏策略。

针对问题三，求解多名玩家在整体天气已知和整体天气未知仅知当天天气时的一般游戏策略，我们引入**博弈模型**，首先利用问题二中的两张地图的最优路径和次优路径来构建每位玩家的策略集，再以此构建**支付矩阵**，利用**规划模型**算出每位玩家的选择偏好。第五关中天气完全已知，两名玩家需事先制定好行动方案，我们得到的最优策略是**两名玩家均直奔终点，路径不完全相同，其中一名玩家需在途中停留一日以构成时间差**。第六关中天气情况未知，我们首先仿照第四关对三位玩家的风险接受程度进行分类并且构建每位玩家的策略集，同第五关中两两解出每位玩家的选择偏好，最后分别制定了适合不同玩家的起始策略。在游戏的进程中，除了使用第四问中的关键点进行判断外，我们还加入了**纳什平衡点**对于玩家相遇时的情况进行分析。

本文使用的方法比较丰富，但是整体结构简单，且每一问的解答方法都与前一问紧密相扣，例如第二问中天气不定的情况利用了第一问中的算法和蒙特卡罗模拟相结合、第三问中的玩家决策利用了第二问中的玩家风险接受程度分类和关键点决策等。这样使得文章整体连贯性强，方法多而不显得繁琐，将问题进行拆分，利用简单方法的组合解决了较为复杂的综合问题。

**关键字**：回溯法、马尔可夫链、蒙特卡罗方法、博弈模型

# 1 问题背景与重述

## 1.1 背景

穿越沙漠的小游戏中，玩家在起点处使用初始资金购买不超过负重上限的水和食物，凭借地图在天气情况变化的沙漠中穿行，途中可在矿山挖矿补充资金或在村庄购买物资。游戏目标为在规定时间内到达终点，并拥有尽可能多的资金。游戏基本规则如下：

(1) 基本时间单位为天，游戏开始时为第 0 天，玩家位于起点。玩家在规定时间内到达终点则游戏结束，否则游戏失败；

(2) 穿越沙漠需水和食物，最小单位均为箱。玩家每天拥有的物资总量不得超过负重上限。若未达终点而物资耗尽则游戏失败；

(3) 每天的天气为“晴朗”、“高温”、“沙暴”三种情况之一，各区域的天气相同；

(4) 每天玩家可从地图中某一区域到达相邻的另一区域（相邻指两区域拥有公共边界，仅有公共顶点不视为相邻），也可选择在原地停留。沙暴天气必须在原地停留；

(5) 玩家在原地停留一天消耗的物资量称为基础消耗量，不同天气基础消耗量不同，行走一天的物资消耗量为基础消耗量的 2 倍；

(6) 玩家在游戏第 0 天时可在起点处用初始资金以基准价格购买水和食物。游戏过程中，玩家可在起点处经过或停留，但不能多次在起点购买物资。玩家到达终点后可退回剩余的水和食物，每箱退回价格为基准价格的一半；

(7) 玩家在矿山停留时可以选择挖矿来获得资金，挖矿一天获得的资金量称为基础收益。如果挖矿，消耗的物资数量为基础消耗量的 3 倍，如果不挖矿，消耗的物资数量为基础消耗量。到达矿山当天不能挖矿，沙暴日可以挖矿；

(8) 玩家经过或停留在村庄时可用剩余的初始资金或挖矿获得的资金购买物资，每箱价格为基准价格的 2 倍。

## 1.2 需要解决的问题

问题一：

假设只有一名玩家，整个游戏过程中每天天气状况事先全部已知，试给出一般情况下玩家的最优策略，求解附件中的“第一关”和“第二关”。

问题二：

假设只有一名玩家，玩家仅知道当天的天气状况，可据此决定当天的行动方案，试给出一般情况下玩家的最优策略，并对附件中的“第三关”和“第四关”进行具体讨论。

问题三：

现有 n 名玩家，他们有相同的初始资金 1，且同时从起点出发。若某天其中任意 $k(2 \leq k \leq n)$ 名玩家均从区域 A 行走到区域 B($B \neq A$), 则他们中的任一位消耗的物资数量均为基础消耗量的 $2k$ 倍；若某天其中任意 $k(2 \leq k \leq n)$ 名玩家在同一矿山挖矿，则他们中的任一位消耗物资数量均为基础消耗量的 3 倍，且每名玩家一天可通过挖矿获得的资金是基础收益的 $\frac{1}{k}$；若某天其中任意 $k(2 \leq k \leq n)$ 名玩家在同一村庄购买物资，每箱价格均为基准价格的 4 倍。其他情况下消耗物资数量与物资价格与单人游戏相同。

（1）假设在整个游戏时段内每天天气状况事先全部已知，每名玩家的行动方案需在第 0 天确定且此后不能更改。试给出一般情况下玩家应采取的策略，并对附件中的“第五关”进行具体讨论。

（2）假设所有玩家仅知道当天的天气状况，从第 1 天起，每名玩家在当天行动结束后均知道其余玩家当天的行动方案和剩余的物资数量，随后确定各自第二天的行动方案。试给出一般情况下玩家应采取的策略，并对附件中的“第六关”进行具体讨论。

# 2 模型假设

1. 假设沙漠中天气不存在突变，即一天内的天气保持一致。
2. 假设玩家均为理性人，行动时均会采取最优策略。
3. 允许玩家到达村庄时物资剩余量为 0。
4. 假设玩家负重不会对物资的消耗产生影响。

# 3 符号说明

| 符号 | 意义 |
|---|---|
| $R_{water}(i,j)$ | 第 i 天位于节点 j 处的水资源剩余量 |
| $R_{food}(i,j)$ | 第 i 天位于节点 j 处的食物剩余量 |
| $C_{water}(i)$ | 第 i 天水资源基础消耗量 |
| $C_{food}(i)$ | 第 i 天食物基础消耗量 |
| $R_{money}(i,j)$ | 第 i 天位于节点 j 处的资金剩余量 |
| $E_{money}$ | 挖矿一天获得的资金基础收益 |
| $C_{money}(i)$ | 第 i 天资金净收入 |
| $T_0$ | 游戏结束时的天数 |

# 4 问题分析

## 4.1 问题一分析

问题一只有一名玩家，且全程的天气已知，我们可以凭此写出每天物资剩余量和资金剩余量的递推表达式以及玩家需要满足的约束条件，然后以终点处资金最多为优化目标建立单目标优化模型。简单对地图进行分析可以初步发现两种方案，即直接走到终点减小物资消耗和去矿山挖矿赚取资金。对于不挖矿的方案, 我们直接选择从起点到终点的最短路径, 对于挖矿的方案, 以上两种情况，我们使用用回溯算法分析求解，最后取资金剩余量更高的解。

## 4.2 问题二分析

问题二中仅知道当天天气，但天气的概率分布符合马尔可夫链的特征，我们以问题一中天气情况为已知样本, 得到天气状态转移矩阵，并取极限得到稳定状态概率。第三关不存在沙暴天气，可以直接结合期望模型和最短路径进行计算，第四关利用蒙特卡罗算法模拟，得到不同风险类型的玩家在出发时的最优决策，在关键点利用各约束条件和期望值绘制决策树。

## 4.3 问题三分析

问题三涉及到了多名玩家，是一个博弈论问题，但其主体部分仍和问题一、二非常相似，我们首先要求出每名玩家可以采取的最优路线和次优路线，利用规划模型求出他们的选择偏好，得到他们在起点处的决策方案。但是对于第六关，玩家的方案要根据天气动态决定，并结合关键点决策和两位玩家相遇时的纳什平衡综合考虑。

# 5 模型建立与求解

## 5.1 问题一模型的建立

问题一需要求解已知天气情况下单一玩家的最佳策略，我们分析判断游戏过程中一些选择的优劣性 (如停留地点的选择, 物资的购买等)，建立物资剩余量及资金剩余量的递推关系式。由于游戏目标为在规定时间内完成游戏并获得尽可能多的资金剩余量，因此我们以规定时间内完成游戏为约束条件，以资金剩余量最大化为优化目标，建立单目标优化模型。

### 5.1.1 停留地点的选择

策略 1: 对于矿山以外的区域而言，除沙暴天气要求必须停留外，高温和晴朗天气均不应停留，否则会带来不必要的物资与时间损失。

例如有连续两天天气分别为高温和晴朗，若此时选择直接在高温天行走，花费的物资转化为资金的消耗为 $C_{money} = 2 \times (8 \times 5 + 6 \times 10) = 200$ 元，而选择高温天停留等到晴天再行进的花费为 $C_{money} = (8 \times 5 + 6 \times 10) + 2 \times (5 \times 5 + 7 \times 10) = 290$ 元，高于直接行进，原因是高温天气行进相比较于晴天行进的费用差距仅为 10 元，而任意一天的停留费用都远高于 10 元，从资金上来说，在矿山外地带停留必定会增大亏损，另外也会增加时间损失，因此不考虑在矿区以外的地区停留。

对于矿山而言，玩家可以挖矿来获得资金。考虑到极端情况，即玩家在沙暴天气下挖矿，并且当日所用物资全部在村庄购买，此时挖矿净收入为 $C_{money} = 1000 - 3 \times 2 \times (10 \times 5 + 10 \times 10) = 100$ 元，即极端情况下挖矿仍有正向收益，所以挖矿必定能获得收益，玩家在矿山应尽量多挖矿。

### 5.1.2 物资购买

策略 2: 在起点和村庄购买恰当的物资, 使终点处物资剩余量为零.

由于终点处剩余物资会以基准价格的一半返还到玩家资金，多余物资会造成资金损失，应使终点处物资剩余量为 0。

物资剩余量的递推关系式为:

$$\begin{cases} R_{water}(i+1, j_2) = R_{water}(i, j_1) - sign(i) \cdot C_{water}(i) \\ R_{food}(i+1, j_2) = R_{food}(i, j_1) - sign(i) \cdot C_{food}(i) \end{cases} \tag{1}$$

其中 i 表示游戏天数，$j_1$ 与 $j_2$ 表示相邻的区域。sign(i) 表示第 i 天玩家活动状态，当玩家停留时 $sign(i) = 1$，当玩家行走时 $sign(i) = 2$，当玩家挖矿时 $sign(i) = 3$。整个游戏过程中必须保证物资剩余量大于等于 0，且终点处物资剩余量为 0，则有

$$\begin{cases} R_{water}(T_0, end) = 0 \\ R_{food}(T_0, end) = 0 \end{cases} \tag{2}$$

策略 3: 在起点处购买足够达到村庄的水和食物，多余的负重量全部用来购买食物。

由于村庄的物资价格为基准价格的 2 倍，因此应尽量在起点处购买足够的物资。另外，由于相较于水而言食物的价格更加昂贵且重量更低，应当尽量在起点处购买足够达到村庄的水和食物，多余的负重量全部用来购买食物。

记起点处购买水 $P_{water}$ 箱，食物 $P_{food}$ 箱，资金剩余量为

$$R_{money}(0,1) = 10000 - (5P_{water} + 10P_{food}) \tag{3}$$

在矿山挖矿时会获得资金收益，挖矿时每日资金剩余量递推式为

$$R_{money}(i+1,j) = R_{money}(i,j) + E_{money} \tag{4}$$

记商店处购买物资水 $P_{water}{}'$ 箱，食物 $P_{food}{}'$ 箱，商店购物不需要额外停留，资金剩余量为

$$R_{money}(i,j) = R_{money}(i,j) - (10P_{water}{}' + 20P_{food}{}') \tag{5}$$

其余节点处资金剩余量不会发生变化。

### 5.1.3 单目标优化模型的建立

由于游戏目标为规定时间内到达终点且有尽量多的剩余资金，因此我们建立一个以剩余资金最大化为优化目标，以完成游戏为约束条件的单目标优化模型。

$$max \quad R_{money}(T_0, end)$$

$$s.t. \begin{cases} T_0 \le T_{ruled} \\ R_{water}(i,j) \ge 0 \\ R_{food}(i,j) \ge 0 \\ M_{water}(i,j) + M_{food}(i,j) \le M \\ R_{water}(T_0, end) = 0 \\ R_{food}(T_0, end) = 0 \end{cases} \tag{6}$$

其中 ($T_0$,end) 中，$T_0$ 表示游戏结束时的天数，end 表示终点;(i,j) 表示第 i 天位于节点 j 的中间过程量，其中 $i = 1, 2, ..., T_0$,$j = 1, ..., end$;$M_{water}$ 表示水的重量，$M_{water}$ 表示水的重量,$M_{food}$ 表示食物的重量,M 表示最大负重;$T_{ruled}$ 表示规定时间。式中各增量按前几小节中所列表达式进行迭代。

## 5.2 问题一模型的求解

由于游戏目标为在规定时间内到达终点且有尽量多的资金剩余量，因此主要有两种游戏思路，即以生存优先的不挖矿争取最短时间到达终点，和以资金剩余量优先的利用规定时间中赶路以外剩余的时间多挖矿再前往终点。问题一的求解将比较两种思路的资

金剩余量并选择更多的一方，即

$$R_{money}(T_0, end) = max\left\{R_1(T_0, end), R_2(T_0, end)\right\} \tag{7}$$

### 5.2.1 地图的简化

首先利用弗洛伊德算法对第一关的地图求解最短路径，然后仅保留地图上的关键节点，即起点、终点、村庄和矿山，而将其他普通点简化为步长，可以画出第一关的地图如图 1 所示。

第二关问题的简化同理.

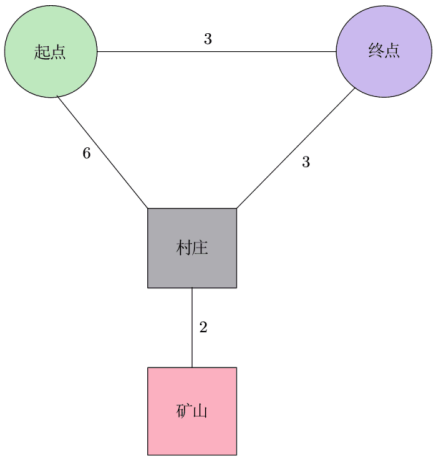


图 1: 第一关最短路线图

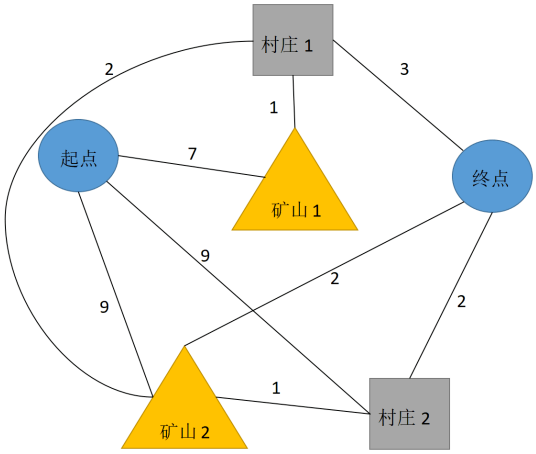


图 2: 第二关最短路线图

### 5.2.2 不挖矿情况

在不挖矿时, 游戏过程中没有资金正向收益，玩家需要减少资金消耗以保证最大资金剩余量。具体策略如下:

从起点处选择最短路径到达终点，除沙暴必须停留在原地外每天前进到下一个区域;

只在起点处购买消耗物资，且购买量恰好为使用量，到达终点处物资剩余量均为 0。

在 matlab 中，使用弗洛伊德算法获得从起点到终点的最短路径，购买恰好需要的水和食物的量, 第一关和第二关的通关情况如下表:

表 2: 不挖矿情况下第一关和第二关的最优通关情况

| 关卡数 | 第一关 | 第二关 |
|---|---|---|
| 最短路径 | $1 \to 25 \to 26 \to 27$ | $1 \to 9 \to 18 \to 26 \to 35 \to 43 \to 52 \to 60 \to 61 \to 62 \to 63$ |
| 起点处水购买量 | 42 箱 | 182 箱 |
| 起点处食物购买量 | 38 箱 | 170 箱 |
| 最大资金剩余量 | 9410 元 | 7390 元 |

### 5.2.3 挖矿情况下使用回溯算法求最优路径

若参与者选择挖矿, 则可能有多种路线选择方式, 如起点-> 矿山-> 村庄-> 终点, 起点-> 村庄-> 矿山-> 终点等, 甚至可能来回往返村庄, 矿山之间补给物资, 因此, 我们必须考虑所有可能的情况, 并对一些不可能的情况进行剪枝, 以减少遍历次数. 我们设计了回溯法求解. 该算法的精巧之处在于: 考虑到村庄补给的情况较为复杂, 我们设计的算法没有按照常规的路径去考察村庄是否需要补给, 而是在每一个节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.

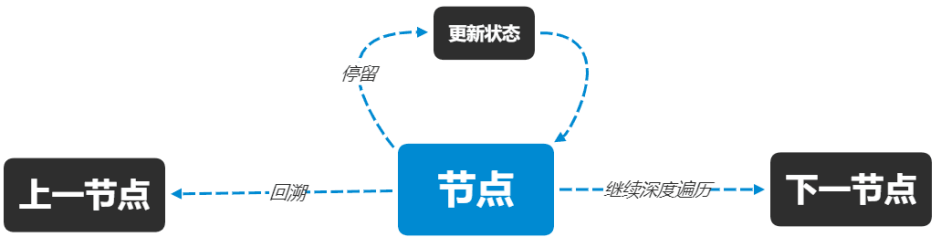


图 3: 节点示意图

算法具体步骤如下:

step 1：只将关键点作为节点，设置起点、村庄、矿山、终点四类节点，并设置算法中各常数，如节点之间距离、基础物资消耗量、每日天气和基准价格等，令 i=0。

step 2：检查 i 是否已经到达参与者的负重上限 M，即 1200 千克，若是则输出结果，算法结束，否则令 i=i+1。

step 3：由前面 5.1.3 分析可知，在起点处采取的策略是将物资买到最大负重量，所以在起点购买 $\frac{i}{3}$ 箱水，$\frac{M-i}{2}$ 箱食物。若没有考察过 $(\frac{i}{3}, \frac{M-i}{2})$ 的情况则继续进行 step 4,

进行深度遍历，否则回到 step 2。

step 4：若当前情况下，该节点为终点，考虑更新答案以及相应的答案数组，若遍历时间超过了截止时间，则此次遍历失败，进行回溯。

step 5：若当前情况下，该节点为村庄，更新玩家所携带的生活物资数量。

step 6：若当前情况下，该节点为矿山，则考虑挖矿和不挖矿仅停留两种情况，若不挖矿仅停留，则检查该节点处物资是否缺乏，若是则回溯到上一个村庄，否则就减去消耗的物资的数量；若挖矿，则在以上基础上，如果物资不缺乏时还需加上挖矿所得的收益。

step 7：每次遍历结束后回到 step 3，决定输出结果或进行下一次遍历。

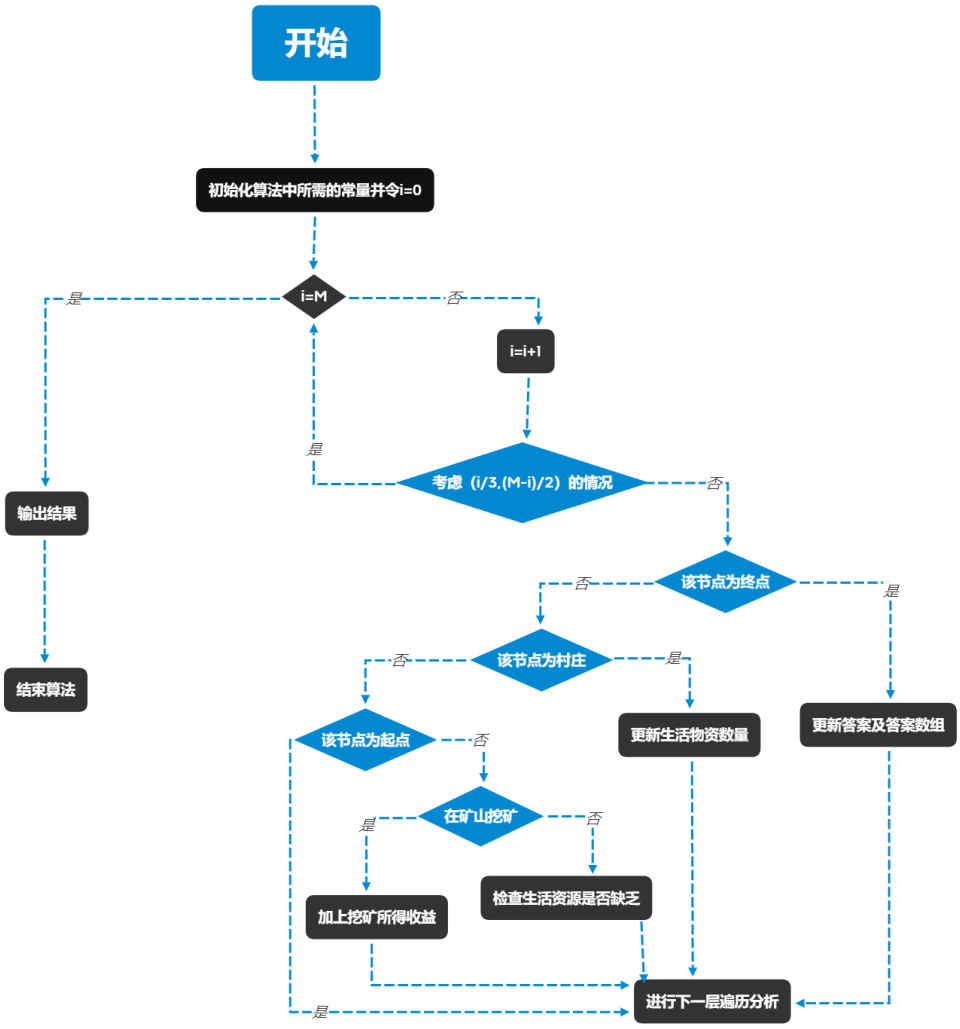


图 4: 回溯算法流程图

使用 matlab 编程求解，求得剩余资金量如表所示：

表 3: 挖矿情况下“第一关”和“第二关”的最优通关情况

| 关卡数 | 第一关 | 第二关 |
|---|---|---|
| 起点处水购买量 | 178 箱 | 130 箱 |
| 起点处食物购买量 | 333 箱 | 405 箱 |
| 最大资金剩余量 | 10470 元 | 12730 元 |

#### 5.2.4 第一关结果分析

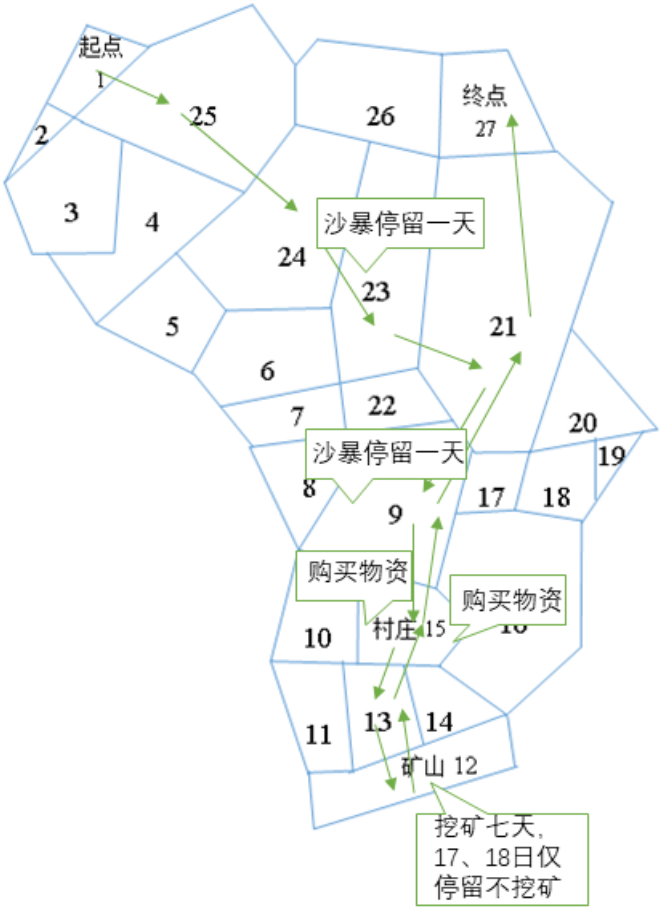


图 5: 第一关最优策略图

第一关当中，将不挖矿与挖矿两种游戏思路进行比较，显然挖矿情况下资金剩余量更多。因此对于已知天气单一玩家通过第一关的情况，最优策略为在起点处购买 178 箱水与 333 箱食物，沿 $1 \to 25 \to 24 \to 23 \to 21 \to 9 \to 15 \to 13 \to 12 \to 13 \to 15 \to 9 \to 21 \to 27$ 的路线，除沙暴和矿山外不停留，在矿山共待九天，17、18 日沙暴天气停留但不挖矿，剩余七天挖矿，两次路过村庄均进行水和食物的补充，如图 5 所示，最大的剩余资金量为 10470 元。

### 5.2.5 第二关结果分析

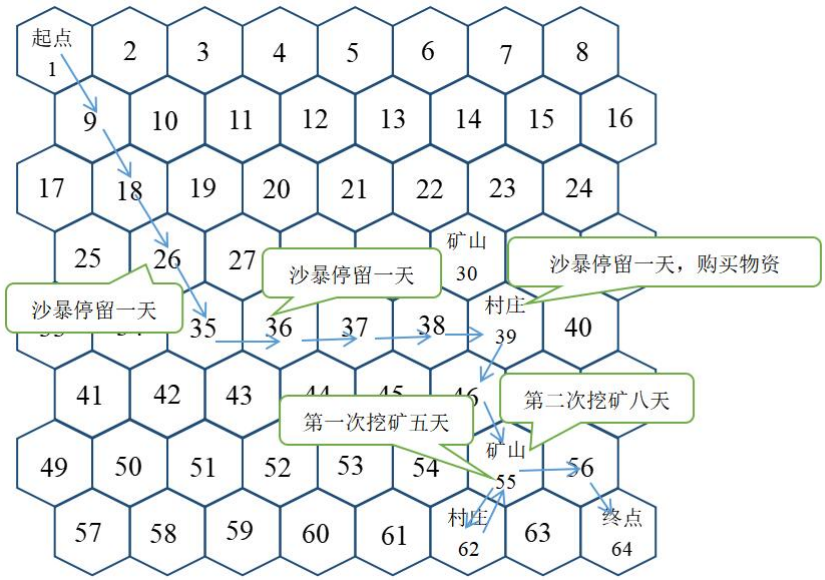


图 6: 第二关最优策略图

从图 6 中可以看出，玩家除了在路上因为沙暴天气而被迫停留外，其他时间均正常行进或挖矿，从 $1 \to 9 \to 18 \to 26 \to 35 \to 36 \to 37 \to 38 \to 39$, 第一次到达村庄 1 进行补给，然后前往矿山 2 挖矿五天，之后去村庄 2 补给后返回挖矿八天，于 30 号到达终点，最大的剩余资金为 12730 元。

详细数值结果见附录或支撑材料 Result.xlsx

## 5.3 问题二模型的建立

问题二和问题一的不同之处在于，玩家每天仅知道当天的天气并以此作出行动规划。我们使用马尔可夫链计算不同天气出现的概率，以此计算影响玩家决策的数学期望，即玩家当日行走所耗物资折合资金消耗期望 $E_1$，玩家当日挖矿所耗水资源箱数期望 $E_2$，玩家当日挖矿所耗食物箱数期望 $E_3$，玩家当日挖矿所耗物资折合资金消耗期望 $E_4$，玩家当日挖矿净资金收入期望 $E_5$，随后建立基于期望的数学模型用于关键点的决策判断，另外采用蒙特卡罗模拟得出不同风险接受程度的玩家在起点应当选取的策略。

### 5.3.1 基于马尔可夫链的天气概率预测

马尔可夫链由满足马尔可夫性质的转移概率分布组成，其特点在于无记忆性，即下一刻的状态只与当前状态有关，而与之前全部状态无关。具备该性质的离散随机过程即为马尔可夫链。本题背景下天气的状况可视为离散随机过程，只知当前天气状态预测第

二天天气状态符合无记忆性，符合马尔可夫链的定义。因此可以使用马尔可夫链模拟各种天气的出现概率 $P_i$。

对于 n 种状态组成的概率空间 $\omega$ 中的状态 i 而言，其下一时刻可能出现包括自身在内的 n 种转移状态，即 $i \to 1$, $i \to 2$, ...,$i \to i$, ...,$i \to n$。根据已有样本可以得到每种转移状态对应的出现概率，即状态转移概率，并以此得到状态转移概率矩阵。

相应的状态转移概率公式为

$$P_{ij} = P(i \to j) = P(E_j|E_i) = P(x_{n+1} = j|x_n = i) \tag{8}$$

本题当中的高温、晴朗、沙暴三种天气情况，使用问题一中所给的天气数据得到转移情况如图 7 所示

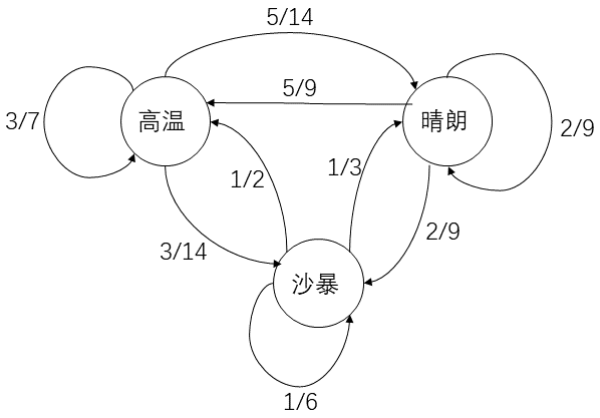


图 7: 原始数据马尔可夫链状态转移概率示意图

获得相应的状态转移概率矩阵为

$$P = \begin{bmatrix} 0.4286 & 0.3571 & 0.2143 \\ 0.5556 & 0.2222 & 0.2222 \\ 0.5000 & 0.3333 & 0.1667 \end{bmatrix} \tag{9}$$

其中 $P_{1j}$ 表示高温向其他天气状况转移，$P_{2j}$ 表示晴朗向其他天气状况转移，$P_{3j}$ 表示沙暴向其他天气状况转移。

经历 k 次天气状态转移后，$k \to +\infty$ 时，状态分布趋于稳定，得到的天气状态转移矩阵为

$$P(\infty) = P^k = \begin{bmatrix} 0.4286 & 0.3571 & 0.2143 \\ 0.5556 & 0.2222 & 0.2222 \\ 0.5000 & 0.3333 & 0.1667 \end{bmatrix}^k \to \begin{bmatrix} 0.4828 & 0.3103 & 0.2069 \\ 0.4828 & 0.3103 & 0.2069 \\ 0.4828 & 0.3103 & 0.2069 \end{bmatrix} \tag{10}$$

于是得到天气分布概率为 P(高温)=0.4828，P(晴朗)=0.3103，P(沙暴)=0.2069。

对于第三关中 10 天内不会出现沙暴的天气情况，和第四关中 30 天内沙暴出现概率较小的天气情况，可在此基础上继续调整马尔可夫链状态转移概率。

### 5.3.2 影响玩家决策因素的期望计算

记事件 A 为“某一天天气为高温”，事件 B 为“某一天天气为晴朗”，事件 C 为“某一天天气为沙暴”。

1、由所给数据可知，天气晴朗时玩家行走每天消耗水 6 箱、食物 8 箱，天气高温时玩家行走每天消耗水 18 箱，食物 18 箱，于是对于玩家第 i 日行走所耗物资折合资金消耗期望 $E_1(i)$，有

$$E_1(i) = (18P(A) + 6P(B)) \cdot Price_{water} + (18P(A) + 8P(B)) \cdot Price_{food} \tag{11}$$

2、由所给数据可知，天气晴朗时玩家挖矿每天消耗水 9 箱、食物 12 箱，天气高温时玩家行走每天消耗水 27 箱，食物 27 箱，于是对于玩家第 i 日挖矿所耗水资源箱数期望 $E_2(i)$，所耗食物箱数期望 $E_3(i)$，有

$$\begin{cases} E_2(i) = 27P(A) + 9P(B) \\ E_3(i) = 27P(A) + 12P(B) \end{cases} \tag{12}$$

3、玩家第 i 日挖矿所耗物资折合资金消耗期望 $E_4(i)$ 为

$$E_4(i) = E_2(i) \cdot Price_{water} + E_3(i) \cdot Price_{food} \tag{13}$$

4、对于玩家第 i 日挖矿净资金收入期望 $E_5(i)$ 而言

$$E_5(i) = E_{money} - E_4(i) \tag{14}$$

式中 $E_{money}$ 是挖矿一天获得的资金基础收益。

### 5.3.3 关键点决策树

对于关键节点的路径选择，我们依据 5.3.2 中的期望计算，将日期和物资剩余量作为决策变量, 决策树见图8。

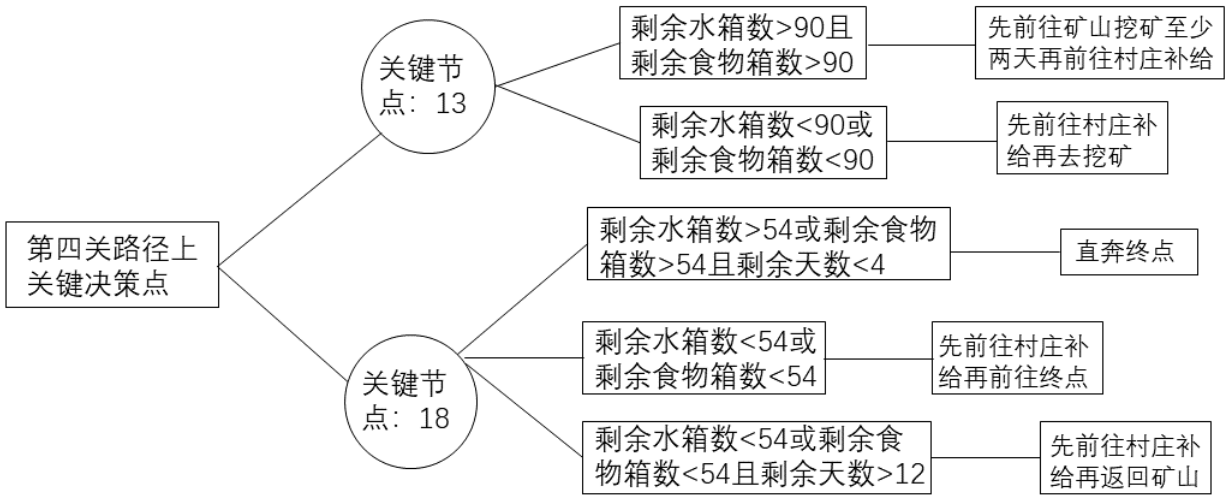


图 8: 第四关关键节点决策选择示意图

## 5.4 问题二模型的求解

当仅知道当天天气情况并据此作出当天行动判断时，问题一中建立的单目标优化模型的求解将变得非常复杂。为了简化问题, 我们使用马尔可夫链预测天气和蒙特卡罗方法模拟，得到在起点处的物资购买情况，对于游戏过程中依据天气的动态决策，则可利用 5.3.3 中的关键点决策树。

### 5.4.1 第三关的一般性策略

第三关的简化地图如图 9 所示.

第三关题设条件有几个重要变化:

(1) 虽然没有给定天气，但是时间只有十天且明确说明了没有沙暴天气，这意味着天气只会导致资金剩余量的不同而不会导致行动路线的改变, 所以仍可以对最优路线进行初步规划。

(2) 第三关的地图上没有村庄进行补给，所以我们首先要保证玩家能够顺利达到终点，必须要根据极端情况计算物资的携带。

(3) 挖矿的基础收益和不同天气的基础消耗量也发生了改变，在当前条件下，在晴天行走的资金消费为 $2 \times (3 \times 5 + 4 \times 10) = 110$ 元而在高温天气行走的消费为 $2 \times (5 \times 9 + 10 \times 9) = 270$ 元，远高于在晴天行走的花费。若出现连续两天一天高温一天晴朗的情况，此时选择在高温天气休息在晴天行走的总花费就会低于在高温天气行走的花费，所以策略中可以考虑在高温天气休息。同时此时在高温天气挖矿的消费为 $3 \times (5 \times 9 + 10 \times 9) = 405$ 元，已经远高于挖矿所得的基本收益 200 元，此时挖矿并不再永远产生正向收益，再加上高温天气的概率较高，因此问题一中的假设需要进行修改。

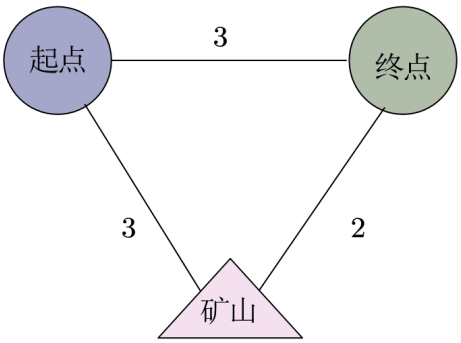


图 9: 第三关最短路线图

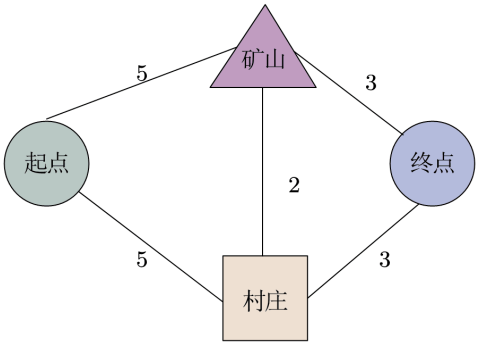


图 10: 第四关最短路线图

### 5.4.2 第四关的一般性策略

第四关挖矿的收益较高，只要挖矿就一定会有正向收益，但是消耗量与第三关一样，即可能存在由于天气原因在矿山外停留的情况。再者由于矿山和村庄距离终点的距离一样，可能会存在往返于矿山和村庄之间补充物资的情况。第四关简化地图如图 10 所示。

### 5.4.3 第三关最优策略与结果分析

由于第三关中游戏规定时间为 10 天，且 10 天内均不出现沙暴天气，调整马尔可夫链转移概率矩阵，得到天气对应概率为 P(晴朗)=0.3889,P(高温)=0.6111。第三关的马尔可夫链概率转移如图 11 所示。

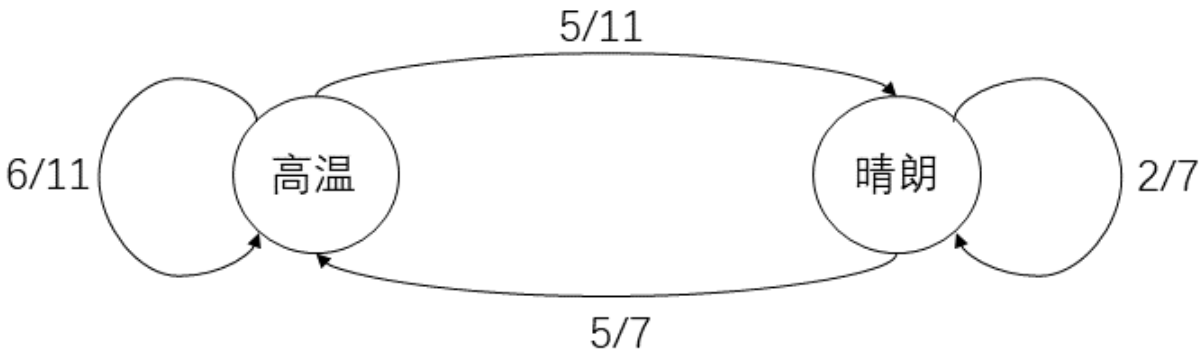


图 11: 第三关马尔可夫链概率转移示意图

使用蒙特卡罗方法多次模拟十日内天气，模拟结果举例如下：

表 4: 使用蒙特卡罗方法得到的多种天气预测组合

| 天数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 模拟天气 *1* | 高温 | 高温 | 晴朗 | 高温 | 晴朗 | 高温 | 高温 | 高温 | 晴朗 | 高温 |
| 模拟天气 *2* | 晴朗 | 晴朗 | 高温 | 高温 | 晴朗 | 晴朗 | 晴朗 | 高温 | 高温 | 高温 |
| 模拟天气 *3* | 高温 | 晴朗 | 高温 | 高温 | 高温 | 高温 | 高温 | 高温 | 高温 | 高温 |
| 模拟天气 *4* | 高温 | 高温 | 晴朗 | 高温 | 晴朗 | 高温 | 晴朗 | 晴朗 | 晴朗 | 高温 |
| 模拟天气 *5* | 高温 | 高温 | 晴朗 | 高温 | 晴朗 | 高温 | 晴朗 | 晴朗 | 晴朗 | 高温 |
| 模拟天气 *6* | 高温 | 晴朗 | 晴朗 | 晴朗 | 晴朗 | 晴朗 | 晴朗 | 晴朗 | 高温 | 高温 |
| 模拟天气 *7* | 高温 | 晴朗 | 高温 | 高温 | 高温 | 高温 | 高温 | 高温 | 高温 | 高温 |
| 模拟天气 *8* | 晴朗 | 高温 | 高温 | 高温 | 高温 | 晴朗 | 晴朗 | 晴朗 | 晴朗 | 高温 |
| 模拟天气 *9* | 高温 | 高温 | 晴朗 | 高温 | 晴朗 | 高温 | 晴朗 | 晴朗 | 晴朗 | 高温 |
| 模拟天气 *10* | 高温 | 晴朗 | 晴朗 | 晴朗 | 晴朗 | 晴朗 | 晴朗 | 晴朗 | 高温 | 高温 |

第三关中通过极端情况计算即可发现，若从最短路径前往挖矿后再到达终点，即全是晴朗，挖矿获得的总收益为 $(200-3\times(3\times5+4\times10))\times5=175$ 元，而从起点直奔终点的路线可以比挖矿少走两天，即全部是晴朗，路上的花费为 $2\times2\times(3\times5+4\times10)=220$ 元，所以第三关选择直奔终点可以使亏损最小化，我们只需要运用马尔可夫链预测天气概率，用第一问中直奔终点的算法计算资金剩余量的期望即可。

### 5.4.4 第四关最优策略与结果分析

(一) 起点处的决策

由于第四关条件为规定时间 30 天内沙暴天气的出现较少，调整马尔可夫链转移概率矩阵，得到天气对应一步转移概率如图 12 所示，天气出现概率为 P(晴朗)=0.3448,P(高温)=0.5518，P(沙暴)=0.1034。

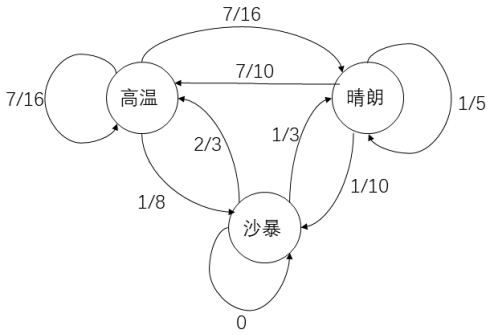


图 12: 第四关马尔可夫链概率转移示意图

根据天气概率情况，使用蒙特卡罗方法模拟 100 次得到天气情况，然后利用回溯法

进行求解，最后对求得结果做统计分析作为起始的决策依据。

蒙特卡罗方法模拟最大资金剩余量结果如图13所示：

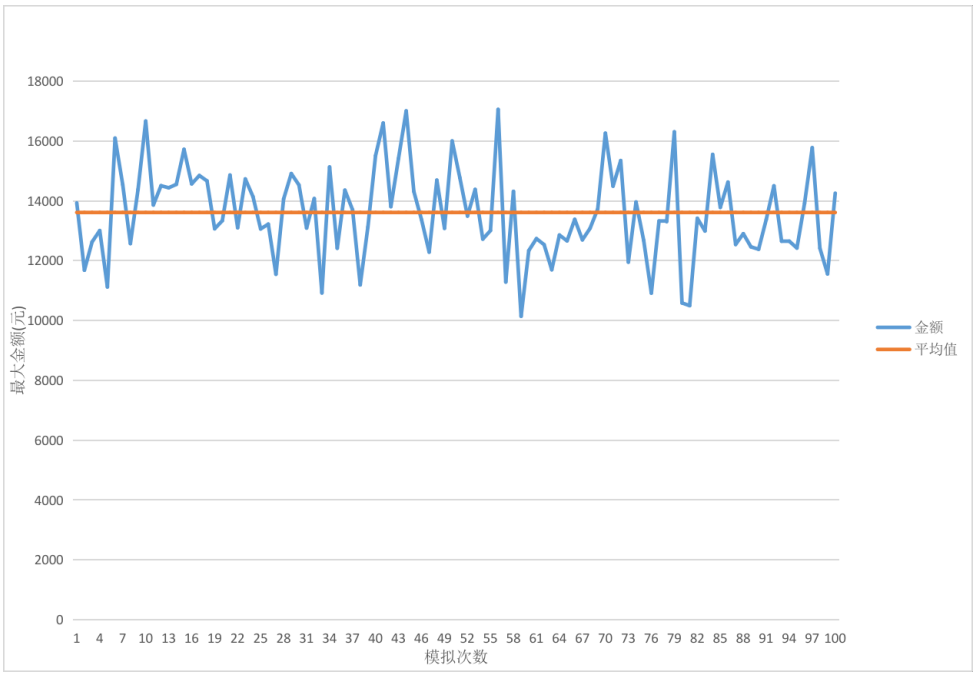


图 13: 第四关蒙特卡罗模拟结果示意图

蒙特卡罗方法仿真模拟的结果显示，玩家的平均资金剩余量为 13610 元。运气更好的玩家可以遇到更多晴朗天气，从而得到更高的挖矿净收益和更少的路途物资消耗，获得更高的资金剩余量，运气较差的玩家可能遇到更多沙暴和高温天气，挖矿净收益较低，物资消耗量较高，所得资金剩余量减少。

我们对蒙特卡罗进行的 100 次模拟结果进行了分类，发现一共有四种路径，路径 1 为起点 → 矿山 → 村庄 → 矿山 → 终点, 路径 2 为起点 → 村庄 → 矿山 → 村庄 → 矿山 → 终点，路径 3 为起点 → 村庄 → 矿山 → 终点，路径 4 为起点 → 矿山 → 终点。在 100 次模拟中，四种路径分别出现了 63 次、19 次、11 次和 7 次。可以发现，这四种策略的差别主要体现在先去村庄还是矿山，是去一次矿山还是两次矿山，这样的结果涉及到了两方面的因素，一方面是在关键点 13、18 玩家所做出的选择，另一方面是玩家在起点处所购买的物资中水和食物的比例，而这点很大程度反应了玩家自身的风险接受程度。

对于玩家的风险接受程度，我们依据玩家的选择倾向将玩家分为风险厌恶型、风险中立型和风险偏好型，对于偏好风险的玩家，我们推荐在一开始选择路径 2，选择多次挖矿且购买大量的食物以减少村庄购买物资消耗的资金；对于厌恶风险的玩家，我们推荐选择路径 4，在起点处购买符合 1:1 比例的食物和水直接前往矿山挖矿后到达终点；对于风险中立型的玩家，我们推荐选择路径 1、3。

## 5.5 问题三模型的建立

在第三问中，玩家从一人增加到多人，由于相同的路径选择会降低收益，玩家不能仅仅考虑最优的路径，还需要考虑其他人的选择，这涉及到博弈论的内容。

首先, 对于出发时物资的购买和初始路线的选择，我们可以根据玩家自身的风险接受程度建立每位玩家的策略集，并以此构建每两名玩家的支付矩阵，求出每位玩家策略集中各个策略的选择概率，为每位玩家推荐最适合自己的策略。其次, 对于在中途遇到其他玩家的情况，我们给出每个节点的纳什平衡点计算，具体解决在不同点与其他玩家相遇时的决策问题。最后, 对于玩家处于关键点时的动态决策，我们采用问题二中构建的决策树进行分析和决策。

### 5.5.1 支付矩阵的构造

每名玩家可以根据自己的喜好和风险接受程度选择不同的路线，不妨假设其中任意两名玩家可以选择的路径数分别为 $n_1$ 和 $n_2$，此时这两名玩家构造的支付矩阵为

$$A_{n_1n_2} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n_2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_11} & a_{n_12} & \cdots & a_{n_1n_2} \end{bmatrix} \tag{15}$$

矩阵中元素 $a_{ij}(i = 1, 2, \cdots n_1; \quad j = 1, 2, \cdots n_2)$ 的意义是第一位玩家选择的 i 路径与第二位玩家选择的 j 路径未冲突的次数与冲突次数之差，其中当两人从同一点出发前往同一地点时视为冲突一次，否则不冲突。因此，冲突越少的路线对于一位玩家来说利益的损失越小，即其选择这条路线的概率更大，所以可以将不冲突次数与冲突次数之差作为支付矩阵中一种方案被选择的概率的体现。

### 5.5.2 策略集与效用期望的构造

假设第一名玩家选择路线 i 的概率为 $p_i(i = 1, 2 \cdots n_1)$，第二名玩家选择路径 j 的概率为 $q_j(j = 1, 2 \cdots n_2)$，则两名玩家构成的策略集为

$$Q_1 = \left\{ \boldsymbol{p} = (p_1, p_2, \cdots, p_{n_1}) \mid 0 \leqslant p_i \leqslant 1, \sum_{i=1}^{n_1} p_i = 1 \right\} \tag{16}$$

$$Q_2 = \left\{ \boldsymbol{q} = (q_1, q_2, \cdots, q_{n_2}) \mid 0 \leqslant q_i \leqslant 1, \sum_{i=1}^{n_2} q_i = 1 \right\} \tag{17}$$

在本题中两名玩家所面对的生存环境和游戏规则均一致，所以他们的效用期望相

等，为

$$U_1(p,q) = U_2(p,q) = \boldsymbol{pAq} = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} p_i a_{ij} q_j \tag{18}$$

对于玩家而言，他们所选择的方案应当使得效用期望最大，即满足

$$\max_{p\in Q_1} U_1(p,q) = \boldsymbol{pMq} \tag{19}$$

$$\max_{q\in Q_2} U_2(p,q) = \boldsymbol{pMq} \tag{20}$$

而在博弈论中，另一方不改变策略的情况下，一方的行为是无法影响最后的结果的，所以两者的决策最后会达到平衡点，这时两人都会在对方所选择方案的条件下，选择自己所获利最大的那一个，而对方选择的方案对自己而言又将是自己获利最小的那一种。即两人都选取满足在对方选取的令自己获利最小方案的条件下，自己获利最大的方案。因此可以将式 (19)、(20) 转化成

$$\max\min \boldsymbol{pA} \tag{21}$$

$$\max\min \boldsymbol{Aq} \tag{22}$$

### 5.5.3 已知天气且事先确定方案的最佳策略模型

若已知全部天气且没有村庄的存在，一开始确定方案之后不能再更改，意味着在起点的时候买刚好合适的物资即可。由于竞争者一共有两人，所以只需要两人在空间上或时间上错开即可，比如两人都选最优路径但一人多停留一天，或一人选择最优路径另一人选择次优路径，但是，停留以错开时间的物资消耗显然要低于选择次优路径行走的物资消耗，所以选择停留一天的方法。由于只有两名玩家，求得的结果与单人时候的最优策略相近。

### 5.5.4 仅知道当天天气且动态确定方案的最佳策略模型

若仅知道当天天气，我们不仅要同时考虑三名玩家，还需要预测之后的天气状况，考虑每位玩家根据当天天气将会作出的决策。

首先利用第四关蒙特卡罗模拟中所得到的的的最优路径和各种路径所对应的风险接受程度，将玩家分类，以此预测在出发时各位玩家的物资购买方式和行动路线，在途中的关键点利用以往天气对于携带食物量的消耗依据决策树对接下来的策略进行判断，最后当遇到其他玩家时，计算两人之间的纳什平衡点来决定下一步的行动方案。

## 5.6 问题三模型的求解

### 5.6.1 已知天气且实现确定方案的最佳策略模型求解

在第五关的求解中，我们首先构造出两名玩家的策略集和支付矩阵，依据第三关中的最优路径 $(1 \to 5 \to 6 \to 13)$，还可以得到两条次优路径 $(1 \to 5 \to 5 \to 6 \to 13)$,$(1 \to 1 \to 1 \to 5 \to 6 \to 13)$。将其分别编号为 3、1、2。对于玩家 1，可以选择路径 1、2、3，而玩家 2 则选择路径 1、2，因此可以得到两人的支付矩阵为:

$$A_{3\times 2} = \begin{bmatrix} -3 & 4 \\ 4 & -3 \\ 1 & 3 \end{bmatrix} \tag{23}$$

其中的元素计算方法如 5.5.1 中一致，例如 $a_{31} = 1$ 是因为玩家 1 所选择的路径 3 和玩家 2 所选择的路径 1 之间有一处冲突两处不冲突，所以得到的值为 1。然后利用 Lingo 编程依据以下算法求出规划模型的全局最优解:

step 1: 为每名玩家建立一个数组，分别记为 $c_1$、$c_2$ 并将其中所有元素初始化为 0。

step 2: 在两位玩家的策略集中，若第一名玩家选择 i 路线，第二名玩家选择 j 路线，则 $c_1$ 中第 i 个元素加 1，$c_2$ 中第 j 个元素加 1。

step 3: 对于任意一名玩家其选择的策略为数组中元素值最大的下标路径，即其选择次数最多的路径。

由此求得的两位玩家的策略集为 $Q_1 = (0, 0.22, 0.78)$、$Q_2 = (0.5, 0.5)$，所以玩家 1 应当选择路径 3，即第三问中的最优路径，而玩家 2 应当选择路径 1 或 2，可以视当天的天气情况而定，这两种决策都接近第三问中的最优路线，只是增加了停留的时间。

### 5.6.2 第五关的一般性策略

第五关和第三关的地图和挖矿收益一致，因此在选择路线时直接从起点前往终点的收益会高于去挖矿的收益，所以最优路线为直接前往终点，次优方案是为了避免冲突而前往挖矿、适当绕路或原地等待，但是绕路的成本显然高于原地等待，所以两人竞争时，可以选择直接前往终点，或在中途停留一天后前往终点，路线示意如图 14 所示:

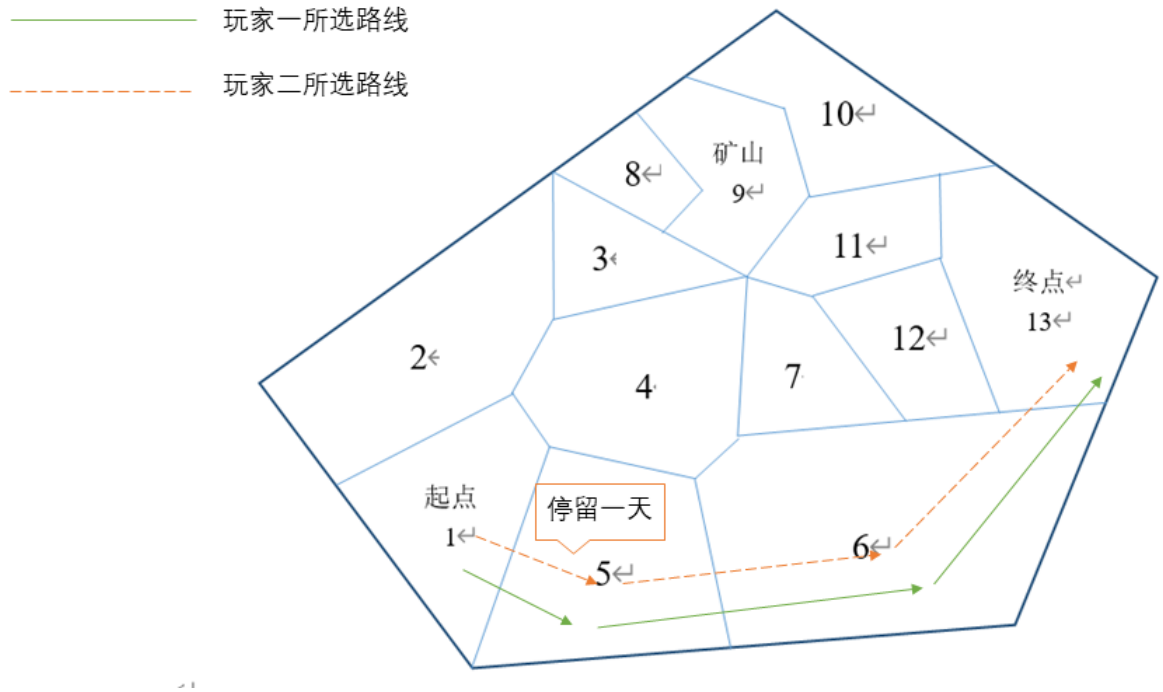


图 14: 第五关最优策略图

### 5.6.3 仅知道当天天气且动态确定方案的最佳策略模型求解

我们使用第四关蒙特卡罗模拟的结果规定最优路径和次优路径，然后利用第五问中的算法，首先对三个玩家两两分组，每组计算一个支付矩阵，并得到两个策略集，然后得到每位玩家各自的策略集并选择相应的方案。

### 5.6.4 开局的策略选择

我们沿用第四关中的四条路径，即路径 1 为起点 → 矿山 → 村庄 → 矿山 → 终点，路径 2 为起点 → 村庄 → 矿山 → 村庄 → 矿山 → 终点，路径 3 为起点 → 村庄 → 矿山 → 终点，路径 4 为起点 → 矿山 → 终点。

我们可以根据玩家的风险接受度对他们分类。在多个玩家相遇时，风险接受度越高的玩家，越有可能选择继续走自己的最优路径，而不管其他其他玩家的选择；风险接受度越低的玩家，越有可能选择停留以避开与其他玩家选择相同的路径。假设三名玩家分别为低风险接受度玩家、高风险接受度玩家和中风险接受度玩家，玩家 1 选择路径 1、3、4，玩家 2 选择路径 1、2、3，玩家 3 选择 1、3。计算支付矩阵和策略集分别为：

玩家 1、2 的支付矩阵：

$$A_{3\times3}=\begin{bmatrix}-12 & 4 & 2\\ 2 & -4 & -10\\ -2 & 0 & 0\end{bmatrix} \tag{24}$$

算出的玩家 1、2 的策略集为 $Q_1=(0,0,1)$、$Q_2=(0,0.71,0.29)$

玩家 1、3 的支付矩阵：

$$A_{3\times2}=\begin{bmatrix}-12 & 2\\ 2 & -10\\ -2 & 0\end{bmatrix} \tag{25}$$

算出的玩家 1、3 的策略集为 $Q_1=(0,1,0)$、$Q_3=(0,1)$

玩家 2、3 的支付矩阵：

$$A_{3\times2}=\begin{bmatrix}-12 & 4\\ 4 & -14\\ 2 & -4\end{bmatrix} \tag{26}$$

算出的玩家 2、3 的策略集为 $Q_2=(0.14,0,0.86)$、$Q_3=(0.25,0.75)$

因此, 开局时的策略应该是玩家 1 选择路径 3 或 4，具体情况根据第四节中关键节点 13 的判断；玩家 2 选择 2 或 3，具体情况根据第四问中关键节点 18 的判断；玩家 3 选择路径 3。

### 5.6.5 在不同地点遇到其他玩家时的选择

当玩家在不同地点遇到其他玩家时，通过计算纳什平衡，决定他应当选择的策略。

(一) 在矿山遇到其他玩家

如果两名玩家同时抵达矿山而且水和食物均充足，且假设天气为高温，则两名玩家的资金变化情况如表 5 所示。

可以看到，无论另一位玩家如何选择，每位玩家的最优选择都是挖矿，两人均挖矿为当前条件下的纳什平衡点。

表 5：矿山两名玩家策略分析

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

(二) 在高温天气遇到其他玩家

在第六关中高温天气行走的消耗本就很高，如果此时还与其他玩家一起使消耗翻倍，那么就远不如原地停留，假设天气均为高温，且每名玩家停留后有 50% 的概率行走，此时两人若都停留则可以算出资金亏损为 585 元，其余情况以此类推。则在该情况下两者的资金变化如表 6 所示。

表 6：高温停留两名玩家策略分析

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

所以此时一人行走一人停留为纳什平衡点，如果可以交流的情况下，可以让物资剩余量高的玩家停留而物资剩余量少的玩家行走，如果两人不能进行交流，可以用混合平衡条件求解，即

$$-585P_1 - 405P_2 = -270P_1 - 540P_2 \tag{27}$$

$$P_1 + P_2 = 1 \tag{28}$$

解得 $P_1 = 0.7, P_2 = 0.3$，即两位玩家有更大的概率选择行走。

(三) 在村庄遇到其他玩家

从前几关的分析可以发现，在未知天气的情况下，玩家会购买更多的水和食物保证存活，因此在抵达村庄的时候往往还留有一些水和食物，下面对两个玩家同时抵达村庄的情况进行分析。

假设两人在起点处都购买基础价格为 650 元的物资，且天气为高温，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资金变化量为-3140 元；一人购买行走，一人停留的话，先离开村庄的人的资金变化量为-650×2-270+595=-975 元，后离开的人的资金变化量为-1705 元。对于两人都停留的情况，假设两人都停留后都以 50%的概率选择购买离开或者继续停留。设每人的资金变化量为 x，那么有：

$$1/4[(-135-x)-975-1705-3140]=-x \tag{29}$$

求得 x=1985，此时的购买策略如表 7 所示，利用混合策略模型，解得停留与购买的概率分别为 55.6% 和 44.4%。所以同时到达的情况下每名玩家都有更大的概率先选择停留。

表 7：村庄两名玩家策略分析

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

### 5.6.6 第六关的一般性策略

考虑到每位玩家对于风险的接受程度不同，在起点处购买物资和选择路线时将根据各自的风险接受程度选择相应的策略集，但是由于天气是不能提前预测的，玩家需要根据天气动态制定方案，此时最重要的就是在两个关键点 (即关卡六地图上的 13,18) 根据自身物资做出行动决策，以及在遇到了其他玩家时根据纳什平衡点得出各自的决策路线。

# 6 模型评价与改进

## 6.1 问题一模型的评价与改进

在问题一中，我们考虑的是单一玩家在天气情况事先完全已知时的最优策略，建立了单目标优化模型，分不挖矿与挖矿两种游戏思路进行讨论，最后得到了第一关与第二

关中均为挖矿路线资金剩余量更高。游戏思路的区分让我们可以更有针对性的选择算法分别处理，降低了深度搜索的遍历时间与剪枝难度，从而更快得到最优策略与理想结果。这种方法也具有很好的普遍性和应用性。

模型一的优点是使用回溯算法大量减少了遍历中的无意义分支运算，使用 $C++$ 进行编程，极大的提高了运算速度，缩短了运行时间，也为问题二和问题三中整体天气情况未知时蒙特卡罗方法的使用提供了可能。但当地图复杂度上升，例如村庄和矿山数量增加、最短路径加长时，模型一的运行时间会指数级增长，这也是深度遍历算法难以回避的问题，可以进一步与动态规划算法结合改进。

## 6.2 问题二模型的评价与改进

在问题二中，我们考虑的是单一玩家在整体天气情况未知、仅知当天天气情况时的最优策略。我们在问题一模型的基础上，进一步考虑了不同关键点处影响玩家决策的关键量的期望，构建了决策树模型，并且使用马尔可夫链对天气概率情况进行计算，并以此为基础使用蒙特卡罗方法多次模拟整体天气情况来仿真实际游戏环境，分别给出了初始位置的决策方法和动态决策方法。

综合全部蒙特卡罗仿真模拟结果，所给出的最佳策略在不同天气环境下具有一般性，模型可以很好的满足问题二要求。但由于计算机性能与时间所限，我们仅进行了 100 次蒙特卡罗仿真模拟，理论上需要更多次模拟以得到更好的结果。可以考虑用更多时间或性能更好的计算机来进一步模拟，继续提高所得结果的精度。另外对于我们采用的蒙特卡洛模拟和决策树相结合的算法虽然较为简单但是其严谨性不如决策模型。

## 6.3 问题三模型的评价与改进

在问题三中，我们考虑的是多个玩家的最优策略，分别为两个玩家在天气情况整体完全已知时的最优策略，和三个玩家在仅知当天天气情况时的最优策略。我们使用博弈模型来处理多玩家游戏问题，并由此得到了不同风险接受程度的每位玩家利益最大化的游戏策略。问题三非常复杂，但可以在逐步拆分后，使用博弈模型结合前两问中已经建立的模型组合求解。

我们为问题三建立的模型可以解决多玩家游戏问题，但具体的博弈过程可以进一步改进，例如加入更多不同风险偏好的玩家和更加丰富的策略集。另外这个方法对人数较少时比较适用，当人数增长时，计算的复杂程度将会大大增加，可能需要考虑其他算法。

# 7　模型的应用与推广

本模型为沙漠游戏而建立，但是对于这一类型的决策游戏都可以适用。同时每一问中使用的具体方法可以推广到更多的实际应用中。例如本文使用的回溯算法可以优化部分深度遍历算法的运算速度，马尔可夫链对天气的预测可以应用到实际动态天气即时决策中，蒙特卡罗仿真模拟辅助决策的方法和博弈模型的结合在现实情况中对于多对象合作与竞争也有广泛应用。

# 参考文献

[1] 戴玲, 夏学知. 基于 Markov 天气模型的流量管理改航策略 [J]. 舰船电子工程,2007(04):57-59+5.

[2] 尚金成, 黄永皓, 张维存, 刘洪杰, 刘文茂. 一种基于博弈论的发电商竞价策略模型与算法 [J]. 电力系统自动化,2002(09):12-15.

[3] 武涛. 博弈论在数学中的应用 [J]. 纳税,2017(18):167.

[4] 胡静. 博弈论在数学中的应用 [J]. 商情 (教育经济研究),2008(05):213+330.

# A 附录：支撑材料

## A.1 第一关、第二关具体数值结果 (即文件 Result.xlsx)

| 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 | 21 | 5780 | 116 | 271 |
| 6 | 9 | 5780 | 100 | 259 |
| 7 | 9 | 5780 | 90 | 249 |
| 8 | 15 | 4150 | 243 | 235 |
| 9 | 13 | 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 | 8150 | 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 | 13 | 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 | 5300 | 130 | 405 |
| 1 | 9 | 5300 | 114 | 393 |
| 2 | 18 | 5300 | 98 | 381 |
| 3 | 26 | 5300 | 88 | 367 |
| 4 | 26 | 5300 | 78 | 357 |
| 5 | 35 | 5300 | 68 | 343 |
| 6 | 36 | 5300 | 52 | 331 |
| 7 | 37 | 5300 | 42 | 321 |
| 8 | 37 | 5300 | 32 | 307 |
| 9 | 38 | 5300 | 16 | 295 |
| 10 | 39 | 3460 | 184 | 283 |
| 11 | 39 | 3460 | 174 | 273 |
| 12 | 46 | 3460 | 158 | 261 |
| 13 | 55 | 3460 | 148 | 247 |
| 14 | 55 | 4460 | 124 | 229 |
| 15 | 55 | 5460 | 100 | 211 |
| 16 | 55 | 6460 | 76 | 193 |
| 17 | 55 | 7460 | 46 | 163 |
| 18 | 55 | 8460 | 16 | 133 |
| 19 | 62 | 4730 | 201 | 207 |
| 20 | 55 | 4730 | 185 | 195 |
| 21 | 55 | 5730 | 170 | 174 |
| 22 | 55 | 6730 | 155 | 153 |
| 23 | 55 | 7730 | 131 | 135 |
| 24 | 55 | 8730 | 116 | 114 |
| 25 | 55 | 9730 | 86 | 84 |
| 26 | 55 | 10730 | 62 | 66 |
| 27 | 55 | 11730 | 47 | 45 |
| 28 | 55 | 12730 | 32 | 24 |
| 29 | 56 | 12730 | 16 | 12 |
| 30 | 64 | 12730 | 0 | 0 |

## A.2 第四关蒙特卡罗仿真模拟具体数值结果

见文件 output_file.txt

# B 附录：源程序代码

## B.1 模型一

### B.1.1 第一关经过矿山的最优方案求解的 C++ 代码

```
#include <iostream>
#include <map>
using namespace std;
```

```
map<pair<int,int>,bool>mp;
const int point[4]={0,1,2,3};//起点,村庄,矿山,终点
// 特殊点的距离
const int ...
    my_distance[4][4]={{0,6,8,3},{6,0,2,3},{8,2,0,5},{3,3,5,0}};
// 特殊点是否相邻
const int f[4][4]={{0,1,1,1},{0,0,1,1},{0,1,0,1},{0,0,0,0}};
// 天气
const int weather[30]={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};
// 题目常数
const int  mx=3,my=2;
const int cx=5,cy=10;
// 对应天气的基础消耗,
const int base_x[4]={0,5,8,10};
const int base_y[4]={0,7,6,10};
// 题目中特殊点的数量,
const int n=4;
// 最大质量
const int maxm=1200;
// 初始资金
const int coins=10000;
// 矿山挖矿的基础收益
const int base_gain=1000;
// 截止日期
const int ddl=30;
// 从第d天出发, 从i点到j点的消耗的水,食物,实际天数
int costx[32][4][4];
int costy[32][4][4];
int actual_days[32][4][4];
int opt_coins=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已经
            //其余的数字分别代表当天玩家位于对应的特殊点,对应的情况如qua数组所
int act[32]; // ...
    每一天的特殊行动情况,2代表挖矿,1代表矿山停止行动,0代表在村庄购买
```

```
41 int ansx[32];//最优路径
42
43 // 在村庄购买水和食物的量
44 int add_water[32];
45 int add_food[32];
46
47 // 最优路径上购买水和食物的量
48 int ans_add_water[32];
49 int ans_add_food[32];
50
51
52 int ansact[32];//最优解路径上的行为
53 int ansg,ansh;//最优解对应的初始水和食物的量
54 int g,h;//用于枚举初始水与食物的量
55
56 void dfs(int day,int now,int nm,int c,int x,int y,int type)
57 {
58     act[day]=type;
59     rec[day]=now;
60
61     if(point[now]==3)
62     {
63         // 要乘以1/2吗
64         if(opt_coins<=c+x*cx+y*cy)
65         {
66         ansg=g;
67         ansh=h;
68
69         opt_coins=c+x*cx+y*cy;
70         for(int i=0;i<=ddl;i++)
71             ansx[i]=rec[i];
72         for(int i=0;i<=ddl;i++)
73             ansact[i]=act[i];
74         }
75         act[day]=-1;
76         rec[day]=-1;
77         return;
78     }
79     if(day>=ddl)
80     {
```

```
81          act[day]=-1;
82          rec[day]=-1;
83          return;
84      }
85      if(point[now]==1)
86          nm=maxm-mx*x-my*y;
87      for(int i=0;i<n;i++)
88          if(f[point[now]][point[i]])
89          {
90          int tx=costx[day][now][i];
91          int ty=costy[day][now][i];
92          int ucost=c;
93          int ux,uy;
94          int um=nm;
95          if(x>=tx)
96              ux=x-tx;
97          else
98          {
99              ux=0;
100             ucost-=2*(tx-x)*cx;
101             um-=(tx-x)*mx;
102         }
103
104         if(y>=ty)
105             uy=y-ty;
106         else
107         {
108             uy=0;
109             ucost-=2*(ty-y)*cy;
110             um-=(ty-y)*my;
111         }
112         if(ucost<0||um<0)
113             continue;
114         dfs(day+actual_days[day][now][i],i,um,ucost,ux,uy,0);
115         }
116     if(point[now]==2)
117     {
118         int attday=day;
119         int tx=base_x[weather[attday]];
120         int ty=base_y[weather[attday]];
```

```
121         attday++;
122         if(x>=tx)
123         {
124         x-=tx;
125         tx=0;
126         }
127         else
128         {
129         tx-=x;
130         x=0;
131         }
132         if(y>=ty)
133         {
134         y-=ty;
135         ty=0;
136         }
137         else
138         {
139         ty-=y;
140         y=0;
141         }
142         nm-=tx*mx+ty*my;
143         c-=2*tx*cx+2*ty*cy;
144         if(nm>=0&&c>=0)
145         dfs(attday,now,nm,c,x,y,1);
146
147         attday=day;
148         tx=base_x[weather[attday]]*2;
149         ty=base_y[weather[attday]]*2;
150         attday++;
151         if(x>=tx)
152         {
153         x-=tx;
154         tx=0;
155         }
156         else
157         {
158         tx-=x;
159         x=0;
160         }
```

```
161             if(y>=ty)
162             {
163             y-=ty;
164             ty=0;
165             }
166             else
167             {
168             ty-=y;
169             y=0;
170             }
171             nm-=tx*mx+ty*my;
172             c-=2*tx*cx+2*ty*cy;
173             c+=base_gain;
174             if(nm>=0&&c>=0)
175             dfs(attday,now,nm,c,x,y,2);
176         }
177         rec[day]=-1;
178         act[day]=-1;
179     }
180
181     int main()
182     {
183         for(int d=0;d<=ddl;d++)
184         {
185             rec[d]=-1;
186             act[d]=-1;
187         }
188         // 初始化从第d天开始，从i到j需要消耗的水和食物
189         for(int d=0;d<ddl;d++)
190             for(int i=0;i<n;i++)
191                 for(int j=0;j<n;j++)
192             if(f[point[i]][point[j]])
193             {
194                 int now=0,count=0,sumx=0,sumy=0;
195                 while(count<my_distance[i][j])
196                 {
197                 if(weather[now+d]!=3)
198                 {
199                     count++;
200                     sumx+=2*base_x[weather[now+d]];
```

```
201                     sumy+=2*base_y[weather[now+d]];
202                 }
203                 else
204                 {
205                     sumx+=base_x[weather[now+d]];
206                     sumy+=base_y[weather[now+d]];
207                 }
208                 now++;
209                 // 超过时间
210                 if(now+d>=ddl)
211                     break;
212                 }
213                 // 超过时间,将sumx和sumy设置为无穷大
214                 if(count<my_distance[i][j])
215                 {
216                     sumx=sumy=20000;
217                     now=30;
218                 }
219                 costx[d][i][j]=sumx;
220                 costy[d][i][j]=sumy;
221                 actual_days[d][i][j]=now;
222             }
223         for(int i=0;i<=maxm;i++)
224         {
225             // 枚举水和食物的质量
226             g=i/mx;
227             h=(maxm-i)/my;
228             if(!mp[make_pair(g,h)])
229             dfs(0,0,0,coins-g*cx-h*cy,g,h,-1);
230             mp[make_pair(g,h)]=1;
231         }
232         for(int i=0;i<=ddl;i++)
233             cout<<i<<":"<<ansx[i]<<";"<<ansact[i]<<endl;
234         cout<<endl;
235         cout<<opt_coins<<" "<<ansg<<" "<<ansh<<endl;
236     }
```

### B.1.2 第二关经过矿山的最优路线求解的 C++ 代码

```
    #include <iostream>
#include <map>
using namespace std;
map<pair<int,int>,bool>mp;
const int point[6]={0,2,1,2,1,3};//起点,村庄,矿山,终点
// 特殊点的距离
const int ...
   my_distance[6][6]={{0,7,8,9,9,11},{7,0,1,3,4,4},{8,1,0,2,3,3},{9,3,2,0,1,2}
// 特殊点是否相邻
const int f[4][4]={{0,1,1,1},{0,0,1,1},{0,1,0,1},{0,0,0,0}};
// 天气
const int weather[30]={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};
// 题目常数
const int  mx=3,my=2;
const int cx=5,cy=10;
// 对应天气的基础消耗,
const int base_x[4]={0,5,8,10};
const int base_y[4]={0,7,6,10};
// 题目中特殊点的数量,
const int n=6;
// 最大质量
const int maxm=1200;
// 初始资金
const int coins=10000;
// 矿山挖矿的基础收益
const int base_gain=1000;
// 截止日期
const int ddl=30;
// 从第d天出发, 从i点到j点的消耗的水,食物,实际天数
int costx[32][6][6];
int costy[32][6][6];
int actual_days[32][6][6];
int opt_coins=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已经
39              //其余的数字分别代表当天玩家位于对应的特殊点,对应的情况如qua数组所
40  int act[32]; // ...
        每一天的特殊行动情况,2代表挖矿,1代表矿山停止行动,0代表在村庄购买
41  int ansx[32];//最优路径
42  int ansact[32];//最优解路径上的行为
43  int ansg,ansh;//最优解对应的初始水和食物的量
44  int g,h;//用于枚举初始水与食物的量
45
46  void dfs(int day,int now,int nm,int c,int x,int y,int type)
47  {
48      act[day]=type;
49      rec[day]=now;
50
51      // 终点!
52      if(point[now]==3)
53      {
54          // 要乘以1/2吗
55          if(opt_coins<=c+x*cx+y*cy)
56          {
57          ansg=g;
58          ansh=h;
59
60          opt_coins=c+x*cx+y*cy;
61          for(int i=0;i<=ddl;i++)
62              ansx[i]=rec[i];
63          for(int i=0;i<=ddl;i++)
64              ansact[i]=act[i];
65          }
66          act[day]=-1;
67          rec[day]=-1;
68          return;
69      }
70      if(day>=ddl)
71      {
72          act[day]=-1;
73          rec[day]=-1;
74          return;
75      }
76      if(point[now]==1)
```

```
77          nm=maxm-mx*x-my*y;
78      for(int i=0;i<n;i++)
79          if(f[point[now]][point[i]])
80          {
81          int tx=costx[day][now][i];
82          int ty=costy[day][now][i];
83          int ucost=c;
84          int ux,uy;
85          int um=nm;
86          if(x>=tx)
87              ux=x-tx;
88          else
89          {
90              ux=0;
91              ucost-=2*(tx-x)*cx;
92              um-=(tx-x)*mx;
93          }
94
95          if(y>=ty)
96              uy=y-ty;
97          else
98          {
99              uy=0;
100             ucost-=2*(ty-y)*cy;
101             um-=(ty-y)*my;
102         }
103         if(ucost<0||um<0)
104             continue;
105         dfs(day+actual_days[day][now][i],i,um,ucost,ux,uy,0);
106         }
107         // 矿山
108     if(point[now]==2)
109     {
110         int attday=day;
111         int tx=base_x[weather[attday]];
112         int ty=base_y[weather[attday]];
113         attday++;
114         if(x>=tx)
115         {
116         x-=tx;
```

```
117         tx=0;
118         }
119         else
120         {
121         tx-=x;
122         x=0;
123         }
124         if(y>=ty)
125         {
126         y-=ty;
127         ty=0;
128         }
129         else
130         {
131         ty-=y;
132         y=0;
133         }
134         // TODO ?
135         nm-=tx*mx+ty*my;
136         c-=2*tx*cx+2*ty*cy;
137         if(nm>=0&&c>=0)
138             // 在村庄不挖矿
139             dfs(attday,now,nm,c,x,y,1);
140
141         attday=day;
142         tx=base_x[weather[attday]]*2;
143         ty=base_y[weather[attday]]*2;
144         attday++;
145         if(x>=tx)
146         {
147         x-=tx;
148         tx=0;
149         }
150         else
151         {
152         tx-=x;
153         x=0;
154         }
155         if(y>=ty)
156         {
```

```
157             y-=ty;
158             ty=0;
159             }
160             else
161             {
162             ty-=y;
163             y=0;
164             }
165             nm-=tx*mx+ty*my;
166             c-=2*tx*cx+2*ty*cy;
167             c+=base_gain;
168             if(nm>=0&&c>=0)
169             // 在矿山挖矿
170             dfs(attday,now,nm,c,x,y,2);
171         }
172         // 普通节点
173         rec[day]=-1;
174         act[day]=-1;
175     }
176
177     int main()
178     {
179         for(int d=0;d<=ddl;d++)
180         {
181             rec[d]=-1;
182             act[d]=-1;
183         }
184         // 初始化从第d天开始，从i到j需要消耗的水和食物
185         for(int d=0;d<ddl;d++)
186             for(int i=0;i<n;i++)
187                 for(int j=0;j<n;j++)
188             if(f[point[i]][point[j]])
189             {
190                 int now=0,count=0,sumx=0,sumy=0;
191                 while(count<my_distance[i][j])
192                 {
193                 if(weather[now+d]!=3)
194                 {
195                     count++;
196                     sumx+=2*base_x[weather[now+d]];
```

```
197                 sumy+=2*base_y[weather[now+d]];
198             }
199             else
200             {
201                 sumx+=base_x[weather[now+d]];
202                 sumy+=base_y[weather[now+d]];
203             }
204             now++;
205             // 超过时间
206             if(now+d>=ddl)
207                 break;
208             }
209             // 超过时间,将sumx和sumy设置为无穷大
210             if(count<my_distance[i][j])
211             {
212                 sumx=sumy=20000;
213                 now=30;
214             }
215             costx[d][i][j]=sumx;
216             costy[d][i][j]=sumy;
217             actual_days[d][i][j]=now;
218         }
219     for(int i=0;i<=maxm;i++)
220     {
221         // 枚举水和食物的质量
222         g=i/mx;
223         h=(maxm-i)/my;
224         if(!mp[make_pair(g,h)])
225         dfs(0,0,0,coins-g*cx-h*cy,g,h,-1);
226         mp[make_pair(g,h)]=1;
227     }
228     for(int i=0;i<=ddl;i++)
229         cout<<i<<":"<<ansx[i]<<";"<<ansact[i]<<endl;
230     cout<<endl;
231     cout<<opt_coins<<" "<<ansg<<" "<<ansh<<endl;
232 }
```

## B.2 模型二

### B.2.1 用蒙特卡罗模拟天气对第四关进行仿真模拟的 C++ 代码

```
    #include <iostream>
#include <map>
#include <time.h>
#include <stdlib.h>

using namespace std;

int weather[30] = {0};



map<pair<int,int>,bool>mp;
const int point[4]={0,1,2,3};//起点,村庄,矿山,终点
// 特殊点的距离
const int ...
   my_distance[4][4]={{0,5,5,8},{5,0,2,3},{5,2,0,3},{8,3,3,0}};
// 特殊点是否相邻
const int f[4][4]={{0,1,1,1},{0,0,1,1},{0,1,0,1},{0,0,0,0}};
// 天气
// const int weather[30]={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};
// 题目常数
const int  mx=3,my=2;
const int cx=5,cy=10;
// 对应天气的基础消耗,
const int base_x[4]={0,3,9,10};
const int base_y[4]={0,4,9,10};
// 题目中特殊点的数量,
const int n=4;
// 最大质量
const int maxm=1200;
```

```
35 // 初始资金
36 const int coins=10000;
37 // 矿山挖矿的基础收益
38 const int base_gain=1000;
39 // 截止日期
40 const int ddl=30;
41 // 从第d天出发，从i点到j点的消耗的水,食物,实际天数
42 int costx[32][4][4];
43 int costy[32][4][4];
44 int actual_days[32][4][4];
45 int opt_coins=0;
46 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已经
47              //其余的数字分别代表当天玩家位于对应的特殊点,对应的情况如qua数组所
48 int act[32]; // ...
    每一天的特殊行动情况,2代表挖矿,1代表矿山停止行动,0代表在村庄购买
49 int ansx[32];//最优路径
50 int ansact[32];//最优解路径上的行为
51 int ansg,ansh;//最优解对应的初始水和食物的量
52 int g,h;//用于枚举初始水与食物的量
53
54 void dfs(int day,int now,int nm,int c,int x,int y,int type)
55 {
56     act[day]=type;
57     rec[day]=now;
58
59     // 终点!
60     if(point[now]==3)
61     {
62         // 要乘以1/2吗
63         if(opt_coins<=c+x*cx+y*cy)
64         {
65         ansg=g;
66         ansh=h;
67
68         opt_coins=c+x*cx+y*cy;
69         for(int i=0;i<=ddl;i++)
70             ansx[i]=rec[i];
71         for(int i=0;i<=ddl;i++)
72             ansact[i]=act[i];
```

```
73          }
74          act[day]=-1;
75          rec[day]=-1;
76          return;
77      }
78      if(day>=ddl)
79      {
80          act[day]=-1;
81          rec[day]=-1;
82          return;
83      }
84      if(point[now]==1)
85          nm=maxm-mx*x-my*y;
86      for(int i=0;i<n;i++)
87          if(f[point[now]][point[i]])
88          {
89          int tx=costx[day][now][i];
90          int ty=costy[day][now][i];
91          int ucost=c;
92          int ux,uy;
93          int um=nm;
94          if(x>=tx)
95              ux=x-tx;
96          else
97          {
98              ux=0;
99              ucost-=2*(tx-x)*cx;
100             um-=(tx-x)*mx;
101         }
102
103         if(y>=ty)
104             uy=y-ty;
105         else
106         {
107             uy=0;
108             ucost-=2*(ty-y)*cy;
109             um-=(ty-y)*my;
110         }
111         if(ucost<0||um<0)
112             continue;
```

```
113         dfs(day+actual_days[day][now][i],i,um,ucost,ux,uy,0);
114         }
115         // 矿山
116     if(point[now]==2)
117     {
118         int attday=day;
119         int tx=base_x[weather[attday]];
120         int ty=base_y[weather[attday]];
121         attday++;
122         if(x>=tx)
123         {
124         x-=tx;
125         tx=0;
126         }
127         else
128         {
129         tx-=x;
130         x=0;
131         }
132         if(y>=ty)
133         {
134         y-=ty;
135         ty=0;
136         }
137         else
138         {
139         ty-=y;
140         y=0;
141         }
142         // TODO ?
143         nm-=tx*mx+ty*my;
144         c-=2*tx*cx+2*ty*cy;
145         if(nm>=0&&c>=0)
146             // 在村庄不挖矿
147             dfs(attday,now,nm,c,x,y,1);
148 
149         attday=day;
150         tx=base_x[weather[attday]]*2;
151         ty=base_y[weather[attday]]*2;
152         attday++;
```

```
153             if(x>=tx)
154             {
155             x-=tx;
156             tx=0;
157             }
158             else
159             {
160             tx-=x;
161             x=0;
162             }
163             if(y>=ty)
164             {
165             y-=ty;
166             ty=0;
167             }
168             else
169             {
170             ty-=y;
171             y=0;
172             }
173             nm-=tx*mx+ty*my;
174             c-=2*tx*cx+2*ty*cy;
175             c+=base_gain;
176             if(nm>=0&&c>=0)
177             // 在矿山挖矿
178             dfs(attday,now,nm,c,x,y,2);
179         }
180         // 普通节点
181         rec[day]=-1;
182         act[day]=-1;
183     }
184
185     int main()
186     {
187         srand((unsigned)time(NULL));
188         for(int i = 0; i < 30;i++ ) {
189             double res = rand()/double(RAND_MAX);
190             if (res <= 0.3448)
191                 weather[i] = 1;
192             else if(res >= 0.8966)
```

```
193             weather[i] = 3;
194         else
195             weather[i] = 2;
196     }
197
198     for(int d=0;d<=ddl;d++)
199     {
200         rec[d]=-1;
201         act[d]=-1;
202     }
203     // 初始化从第d天开始，从i到j需要消耗的水和食物
204     for(int d=0;d<ddl;d++)
205         for(int i=0;i<n;i++)
206             for(int j=0;j<n;j++)
207         if(f[point[i]][point[j]])
208         {
209             int now=0,count=0,sumx=0,sumy=0;
210             while(count<my_distance[i][j])
211             {
212             if(weather[now+d]!=3)
213             {
214                 count++;
215                 sumx+=2*base_x[weather[now+d]];
216                 sumy+=2*base_y[weather[now+d]];
217             }
218             else
219             {
220                 sumx+=base_x[weather[now+d]];
221                 sumy+=base_y[weather[now+d]];
222             }
223             now++;
224             // 超过时间
225             if(now+d>=ddl)
226                 break;
227             }
228             // 超过时间,将sumx和sumy设置为无穷大
229             if(count<my_distance[i][j])
230             {
231                 sumx=sumy=20000;
232                 now=30;
```

```
233                 }
234                 costx[d][i][j]=sumx;
235                 costy[d][i][j]=sumy;
236                 actual_days[d][i][j]=now;
237             }
238         for(int i=0;i<=maxm;i++)
239         {
240             // 枚举水和食物的质量
241             g=i/mx;
242             h=(maxm-i)/my;
243             if(!mp[make_pair(g,h)])
244             dfs(0,0,0,coins-g*cx-h*cy,g,h,-1);
245             mp[make_pair(g,h)]=1;
246         }
247         for(int i = 0; i < 30; i++) {
248             cout<<weather[i] << " ";
249         }
250         cout<<endl;
251 
252         for(int i=0;i<=ddl;i++)
253             cout<<i<<":"<<ansx[i]<<";"<<ansact[i]<<endl;
254         cout<<endl;
255         cout<<opt_coins<<" "<<ansg<<" "<<ansh<<endl;
256 }
```

## B.3 模型三

### B.3.1 求解第五关的 Lingo 代码

```
1       model:
2   sets:
3   k/1..3/:s;
4   n/1..2/:t;
5   pay(k,n):M;
6   endsets
7 
8   data:
9   M=-3 4
```

```
10    4 -3
11    1 3;
12  enddata
13
14  !max=a;
15  !@for(n(j):
16      a<@sum(k(i):s(i)*m(i,j));
17  !);
18
19  !@sum(k:s)=1;
20
21  max = a;
22  @for(k(j):
23  a<@sum( n(i):t(i)*m(j,i) );
24  );
25  @sum(n:t)=1;
26  End
```