穿越沙漠

摘要

在穿越沙漠游戏中，玩家凭借一张地图，利用初始资金购买一定数量的水和食物，从起点出发，在沙漠中行走。途中会遇到不同的天气，也可在矿山、村庄补充资金或资源，目标是在规定时间内到达终点，并保留尽可能多的资金。题目基于不同的情况给出了诸多问题，我们针对这些问题，给出以下解决方案：

针对问题一，鉴于所有的天气的信息都是已知且玩家只有一名，故对“第一关”和“第二关”采用动态规划的方法，建立状态转移方程，得到递推关系。由于图本身的规模较大，我们基于几个重要节点 (起点、矿山、村庄、终点) 之间的最短路，对图进行了简化。最后，根据列出的递推关系式对问题进行了求解，得到了“第一关”游戏结束时的最大资金数为 10470 元；“第二关”游戏结束时的最优资金数为 12730 元。

针对问题二，我们需要根据只知道当天天气这个限制，给出玩家通过“第三关”和“第四关”的最优策略。因为整个 30 天的天气未知，所以我们分别设定了晴朗天气、高温天气和沙暴天气发生的概率。在“第三关”中，只有晴朗天气和高温天气，设定晴朗天气的概率在 [0,1] 之间，以 0.05 为间隔；在“第四关中”，设定沙暴天气概率为 0.05，晴朗天气在 [0,1] 之间，以 0.01 为间隔。对于同样的概率，我们进行了大量的数值模拟（1000 次），通过模拟结果，我们发现不管天气分布如何，最终的最优策略都不会改变。在“第三关”中采用了从起点直接到终点的策略，得到游戏结束后的资金数在 9190 到 9430 元之间；在“第四关”中采用了直接去矿山挖矿、只去村庄补给一次的策略，得到游戏结束时的资金期望为 18080 元。

针对问题三，由于玩家之间的行动彼此受到限制，因此我们需要综合考量所有玩家的行动方案。我们设定所有玩家以合作博弈的方式进行游戏，即目标为游戏结束时所有玩家的资金总和最大。在“第五关”中，因为挖矿收益只有 200 元，基于“第三关”的结果，我们最终参用了两名玩家不去挖矿，并且一名玩家直接到达终点、另一名玩家在高温天停留一天再抵达终点的策略，得到游戏结束时资金数为 19180 元。在“第六关”中，我们采用了三名玩家都去矿山挖矿且轮流在矿山挖矿的策略，即一人挖矿直到资源不够，另外两人停留在原地等待挖矿，并且一人挖完后至多去村庄补一次的策略，得到了游戏结束时的资金数可达 32522.89 元。

关键字： 动态规划、计算机仿真、合作博弈、穿越沙漠

目录

一、问题重述 ····· **4**

1.1 问题背景 ····· 4

1.2 问题的提出 ····· 4

二、问题分析 ····· **4**

2.1 问题一分析 ····· 4

2.2 问题二分析 ····· 4

2.3 问题三分析 ····· 5

三、模型的假设 ····· **5**

四、符号说明 ····· **5**

五、问题一的模型建立和求解 ····· **6**

5.1 问题一分析 ····· 6

5.1.1“第一关”分析 ····· 6

5.1.2“第二关”分析 ····· 6

5.2 “第一关”模型的建立 ····· 6

5.3 “第一关”模型的求解 ····· 7

5.4 “第二关”模型的建立 ····· 8

5.5 “第二关”模型的求解 ····· 11

六、问题二的模型建立和求解 ····· **12**

6.1 问题二分析 ····· 12

6.1.1“第三关”分析 ····· 12

6.1.2“第四关”分析 ····· 12

6.2 “第三关”模型的建立 ····· 12

6.2.1策略一 ····· 13

6.2.2策略二 ····· 13

6.2.3策略三 ····· 14

6.2.4策略四 ····· 14

6.3 “关卡三”模型的求解 ····· 15

6.4 “关卡四”模型的建立 · · · · · 15
6.5 ”关卡四”模型的求解 · · · · · 16
七、问题三的模型建立和求解 · · · · · 16
7.1 问题三分析 · · · · · 16
7.1.1“第五关”分析 · · · · · 17
7.1.2“第六关”分析 · · · · · 17
7.2 “第五关”模型的建立 · · · · · 17
7.2.1 策略一 · · · · · 18
7.2.2 策略二 · · · · · 18
7.3 “第五关”模型的求解 · · · · · 19
7.4 “第六关”模型的建立 · · · · · 19
7.4.1 策略 · · · · · 20
7.5 ”第六关”模型的求解 · · · · · 20
八、模型的优缺点 · · · · · 21
8.1 模型的优点 · · · · · 21
8.2 模型的缺点 · · · · · 21
参考文献 · · · · · 21
附录 A 第一问动态规划 · · · · · 22
附录 B “第三关”最优策略模拟代码 · · · · · 26
附录 C “第四关”最优策略模拟代码 · · · · · 31
附录 D “第六关”最优策略模拟代码 · · · · · 39
附录 E 支撑材料列表 · · · · · 52

一、 问题重述

1.1 问题背景

随着游戏的普及，越来越多的游戏出现在人们的眼前，激起了大众的兴趣，其中策略游戏也越来越受大家的喜爱。策略游戏可以锻炼脑力，考验敏捷性，思维性；缓解现实社会对我们的压力，暂时忘却自己的烦恼。为了使游戏体验达到最佳，许多的人利用数学建模给游戏制定策略。基于此，本题设定了一个穿越沙漠的策略游戏，其通关法则为在规定时间内达到终点且保证自己的资源足够支撑自己穿越沙漠，除此之外，还要求玩家尽可能多地保留资金。

1.2 问题的提出

根据游戏的基本规则，需要在不同地图和天气下，解决以下几个问题：

(1) 只有一名玩家，游戏过程中所有天气情况已知，给出通过第一关和第二关在一般情况下的最优策略——在游戏不失败的前提下，尽可能多地保留资金。

(2) 只有一名玩家，且玩家仅知道当天地天气情况并作出行动，给出通过第三关和第三关在一般情况下最优策略——在游戏不失败的前提下，尽可能多地保留资金。

(3) 多名玩家同时开始游戏，在给定条件下，给出通过第五关和第六关的策略。

二、 问题分析

2.1 问题一分析

问题一假设只有一名玩家已知整个游戏时间段内每天天气状况，要求我们根据已经游戏的设定和地图给出通过“第一关”和“第二关”的一般情况下玩家的最优策略。选择最优策略问题本质上是一个动态规划过程，目标函数是某个状态下所持资金的函数，根据题目含义，最优意味着资金函数最大。那么，在给定初始条件后并且列出动态规划方程后，我们就可以通过程序求解。

2.2 问题二分析

问题二中要求我们根据已经游戏的设定和地图给出通过“第三关”和“第四关”的一般情况下玩家的最优策略。在“第三关”和“第四关”中，都假设只有一名玩家，且玩家仅知道当天的天气情况，并要据当天的天气情况作出的行动方案。由于情况的复杂性，我们在考虑过程中对某些情况进行了“剪枝”，最后给出几种策略，然后通过计算机模拟就可以进行求解。

2.3 问题三分析

问题三中是多名玩家一起进行游戏，且初始条件完全相同，但两个人的行动受到彼此的一些限制，因此我们需要综合考量这些玩家的行动方案。由于这些限制的存在，玩家们在游戏过程要想最终取得最多的资金书，需要彼此之间互相合作。那么我们也是先分析不同的情况，给出一些策略和限制，最后通过计算机模拟该过程进行求解。

三、 模型的假设

- 假设多名玩家在游戏过程中是合作关系。

四、 符号说明

符号	意义（单位）
W_i	第 i 天的天气
P	玩家所处位置（普通、村庄或矿山）
$F_{initial}$	玩家在起点位置购买的食物箱数（箱）
$W_{initial}$	玩家在起点位置购买的水箱数（箱）
d_{mine}	从起点到矿山的最短路距离需要的天数（天）
d_{dest}	从矿山到终点的最短路距离需要的天数（天）
d_{rest1}	在矿山位置时距离游戏结束的剩余天数（天）
d_{rest2}	在村庄位置时距离游戏结束的剩余天数（天）
d_{pass}	游戏已经开始的天数（天）
d_{period}	玩家总共的游戏时长（天）
f_{high}	高温天气食物的基础消耗量（箱）
f_{sun}	晴朗天气食物的基础消耗量（箱）
f_{storm}	沙暴天气食物的基础消耗量（箱）
w_{high}	高温天气水的基础消耗量（箱）
w_{sun}	晴朗天气水的基础消耗量（箱）
w_{storm}	沙暴天气水的基础消耗量（箱）
w	总共水的消耗量（箱）
f	总共食物的消耗量（箱）
$wc(W_i)$	第 i 天水的基础消耗量（箱）
$fc(W_i)$	第 i 天食物的基础消耗量（箱）
Bw	在村庄购买的水的质量（箱）

Bf	在村庄购买的食物的质量（箱）
$a(W_i, P, w, f)$	在给定状态下所能获得的最多资金函数
p_{storm}	沙暴天气出现的概率
p_{sun}	晴朗天气出现的概率

五、 问题一的模型建立和求解

5.1 问题一分析

问题一中假设只有一名玩家已知整个游戏时间段内每天天气状况，要求我们根据已经游戏的设定和地图给出通过“第一关”和“第二关”的一般情况下玩家的最优策略。

5.1.1 “第一关”分析

“第一关”中，总共的游戏时间为三十天，玩家需要在三十天内到达终点且不能出现资源完全被消耗的情形。玩家已知三十天的所有天气情况，需要根据给定的条件，使得游戏结束时自己的资金达到最大。根据该问题的陈述，可以采用动态规划（Dynamic programming）的方法——为了求解某个状态所有拥有的最大资金数，我们将其分解成若干个子问题（即，下一天在不同情况下所能获得的最大资金数），最后通过求解子问题得到原问题的解。

5.1.2 “第二关”分析

“第二关”中，总共的游戏时间为三十天，玩家需要在三十天内到达终点且不能出现资源完全被消耗的情形。玩家已知三十天的所有天气情况，需要根据给定的条件，使得游戏结束时自己的资金达到最大。本关卡的建模思想与“第一关”基本一致，这里不再赘述。

5.2 “第一关”模型的建立

首先为了方便我们的建模，要把地图抽象成一个简单的有向图（图 1）。根据上述“第一关”的分析，我们可以列出动态规划的状态方程。

当玩家处于普通位置（非村庄、矿山位置），简记为 $P =$ 普通。

$$a(W_i, P, w, f) = max \begin{Bmatrix} a(W_{i-1}, P, w + wc(W_{i-1}), f + fc(W_{i-1})) \\ a(W_{i-1}, P', w + 2 * wc(W_{i-1}), f + 2 * fc(W_{i-1})) \end{Bmatrix}$$

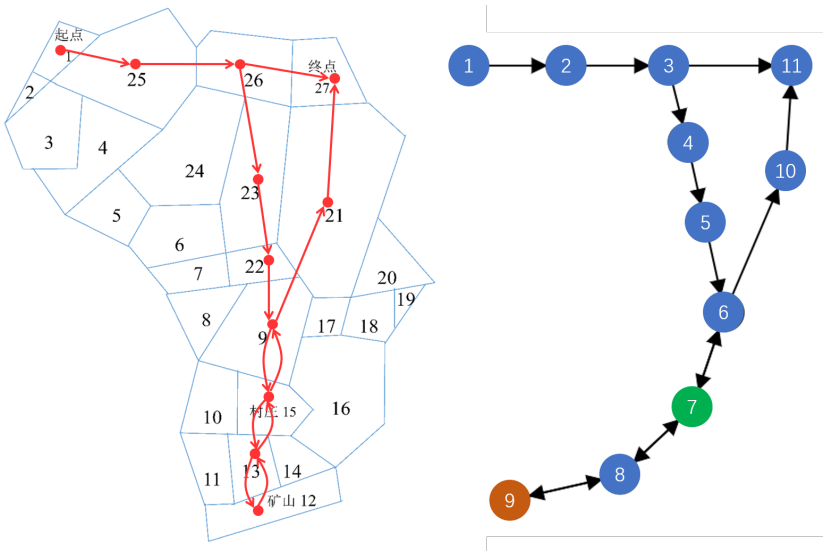

图 **1**　“第一关”有向图

这时，玩家在给定状态能获得的最大资金函数为前一天停留在原地的函数，和前一天从其他位置走过来的函数，两者之间的最大值。

当玩家处于村庄位置，简记为 $P=$村庄。

$$a(W_i,P,w,f)=max\left\{\begin{array}{c}a(W_{i-1},P,w+wc(W_{i-1})-Bw,f+fc(W_{i-1})-Bf)-2(Bw+Bf)\\a(W_{i-1},P',w+2wc(W_{i-1})-Bw,f+2fc(W_{i-1})-Bf)-2(Bw+Bf)\end{array}\right\}$$

这时，玩家在给定状态能获得的最大资金函数为前一天停留在原地并在村庄购买资源后的函数，和前一天从其他位置走过来并在村庄购买资源后的函数，两者之间的最大值。

当玩家处于矿山位置，简记为 $P=$矿山。

$$a(W_i,P,w,f)=max\left\{\begin{array}{c}a(W_{i-1},P,w+wc(W_{i-1})-Bw,f+fc(W_{i-1}))\\a(W_{i-1},P',w+2*wc(W_{i-1})-Bw,f+2*fc(W_{i-1}))\\a(W_{i-1},P,w+3*wc(W_{i-1})-Bw,f+3*fc(W_{i-1}))+C\end{array}\right\}$$

这时，玩家在给定状态能获得的最大资金函数为前一天停留在原地的函数、在原地挖矿的函数，和前一天从其他位置走过来的函数，三者之间的最大值。

状态方程的初始条件为

$$a(W_i,P,w,f)=\begin{cases}10000-5w-10f & W_i=0,P=1\\-\infty & \text{other else}\end{cases}\tag{1}$$

5.3 “第一关”模型的求解

根据以上动态规划的状态方程，我们可以得到以下最优策略：

1 第 0 天，开始购买 178 箱，333 箱食物，共花费 4220 元，剩余 5780 元。
2 第 1-8 天，从起点走到村庄，共消耗 98 箱水，98 箱食物；并在村庄购买 163 箱水，0 箱食物，花费 1630 元，剩余 4150 元。
3 第 9-10 天，从村庄走到矿山，共消耗 32 箱水，24 箱食物。
4 第 11-19 天，在矿场打工，其中 17-18 天在矿场休息两天，其余时间挖去矿藏，共消耗 155 箱水，131 箱食物，并通过挖矿获得 7000 元，剩余 11150 元。
5 第 20-21 天，从矿场走到村庄，共消耗 26 箱水，26 箱食物；并在村庄购买 26 箱水，2 箱食物，共花费 680 元，剩余 10470 元。
6 第 22-24 天，从村庄到终点，共消耗 26 箱食物，26 箱水，此时所剩水和食物皆为 0 箱，剩余 10470 元。

5.4 “第二关”模型的建立

“第二关”与“第一关”思想相同，动态规划的状态方程也相同，但是由于顶点较多，程序解决问题较为困难，故我们将该问题 Q 按情况分为三个子问题 Q_1, Q_2, Q_3，最后以三个子问题结果的最大值为最后的结果，即

$$Q = max\{Q_1, Q_2, Q_3\}$$

Q_1 : 只选择在矿山 30 挖矿，不在矿山 55 挖矿。首先需要找出起点到矿山 30 的最短路，由于村庄 39 离矿山 30 较近，故其挖矿后补给也只会在村庄 39，而不是村庄 62；除此之外，从矿山 30 前进到终点时，由于经过村庄 62 并不是矿山 30 到终点的最短路，故不会前往村庄 62，因此可以将图简化为红色的子图如图 2：

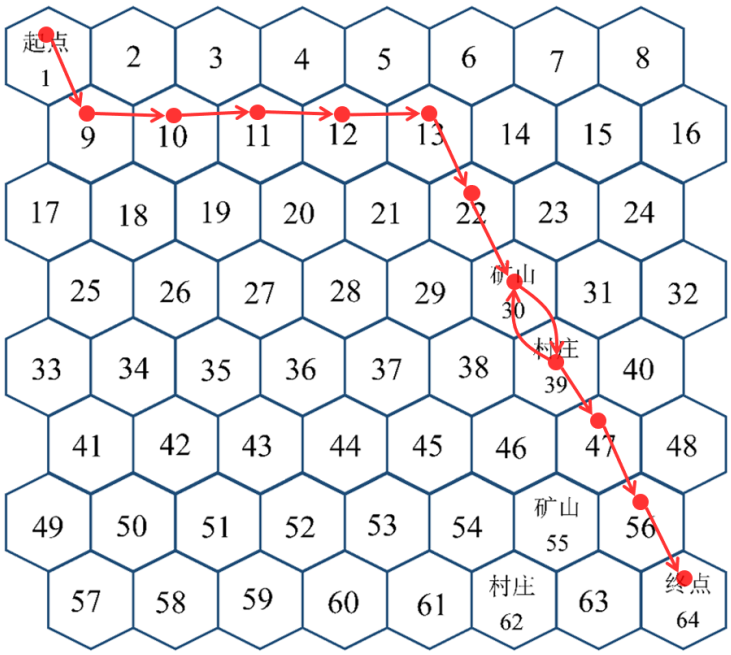

图 **2**　**Q1** 子问题简化图

同时，红色的子图也是不选择挖矿，从起点到终点的最短路。

同第一问一样，我们仍可以使用动态规划在该子图上解决问题。

Q_2 : 只选择在矿山 55 挖矿，不在矿山 30 挖矿。同 Q_1 相同，在找出起点到矿山 55 的最短路后，其挖矿后补给也只会在村庄 62。从矿山 55 前进到终点时，也不会前往村庄 62，因此可以将图简化为红色的子图如图 3：

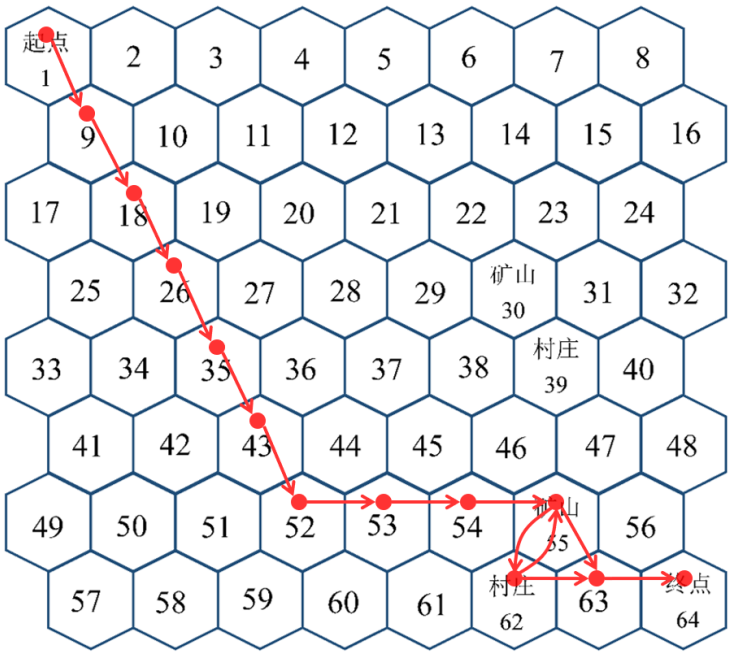

图 **3**　**Q2** 子问题简化图

该问也可以使用动态规划解决。

Q_3：可以前往两个矿山挖矿。由于矿山 30 离起点较近，故在找出起点到矿山 30 的最短路后，由于村庄 39 离其较近，故在矿山 39 挖矿后补给可定于村庄 39。

当要前往矿山 55 挖矿时，找到矿山 30 到矿山 55 的最短路，并令此路经过村庄。又由于村庄 62 离矿山 55 较近，故在矿山 55 挖矿后补给可定于村庄 62。

注意到，到达矿山 55 后，我们不会再返回矿山 30 进行挖矿，因为这会比停在矿山 55 挖矿多消耗路途中的资源，是得不偿失的。那么我们就一定会选择从矿山 55 或村庄 62 前进到终点，找出它们到终点的最短路，原图可以简化为红色的子图如图 4：

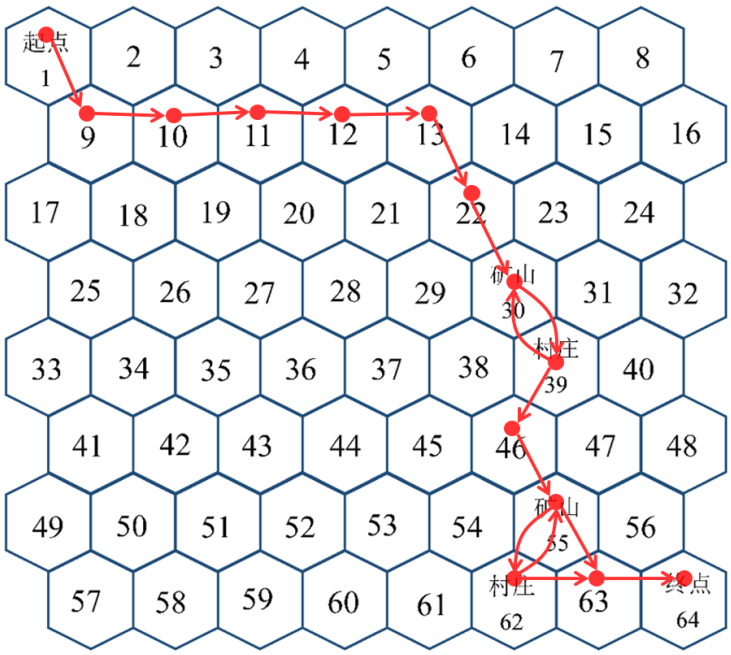

图 **4**　**Q3** 子问题简化图

5.5 “第二关”模型的求解

根据以上分析，可以算出 Q_1、Q_2 、Q_3 问题中所能获得的最大资金分别为 12470，12345，12740，选择 Q_3 子图作为最终的简化图，并且给出其最优策略：

1 第 0 天，开始购买 130 箱，405 箱食物，共花费 4700 元，剩余 5300 元。

2 第 1-10 天：从起点走到村庄 39，共消耗 130 箱水，122 箱食物；并在村庄购买 228 箱水，0 箱食物，花费 2280 元，剩余 3020 元。

3 第 11-13 天：从村庄 39 走到矿山 55，共消耗 36 箱水，36 箱食物。

4 第 14-18 天：在矿山 55 挖矿，共消耗 132 箱水，114 箱食物，并通过挖矿获得 5000 元，剩余 8020 元。

5 第 19 天：从矿场 55 走到村庄 62，共消耗 16 箱水，12 箱食物；并在村庄购买 157 箱水，86 箱食物，共花费 3290 元，剩余 4730 元。

6 第 20 天：从村庄 62 到矿场 55，共消耗 16 箱食物，12 箱水。

7 第 21 天-28 天：在矿山 55 挖矿，共消耗 153 箱水，171 箱食物，并通过挖矿获得 8000 元，剩余 12730 元。

8 第 29-30 天：从矿山 55 到终点，共消耗 32 箱食物，24 箱水，此时所剩水和食物皆为 0 箱，剩余 12730 元。

六、 问题二的模型建立和求解

6.1 问题二分析

问题二中要求我们根据已经游戏的设定和地图给出通过“第三关”和“第四关”的一般情况下玩家的最优策略。在“第三关”和“第四关”中，都假设只有一名玩家，且玩家仅知道当天的天气情况，并要据当天的天气情况作出的行动方案。

6.1.1 “第三关”分析

“第三关”中，总共的游戏时间为十天，玩家需要在十天内到达终点且不能出现资源完全被消耗的情形。由于该地图中没有村庄，玩家需要在起点处带够水和食物这两个资源，且不能用完初始资金也不能超过负重上限。在非矿山位置每天有两种选择——停留和走一步，在矿山位置每天有三种选择——停留、走一步和挖矿。基于情况的复杂性，我们采用 matlab 模拟的方法选择一种最优策略。在制定策略前，我们需要考虑到一些情况经过计算后亏本比较多，我们将其“剪枝”。这样，能够简化需要考虑的情况。

6.1.2 “第四关”分析

“第四关”中，总共的游戏时间为三十天，玩家需要在三十天内到达终点且不能出现资源完全被消耗的情形。玩家只知道当天的天气情况，并且需要根据当天天气作出一些选择，如在非矿山位置每天有两种选择——停留和走一步，在矿山位置每天有三种选择——停留、走一步和挖矿。由于该地图中矿山的收益比较高（挖矿的每天基础收益为 1000 元），要想达到最大收益，玩家需要尽可能多地待在矿山挖矿。除此之外，由于有村庄的存在，玩家需要考量起始位置带的资源数、是否前去村庄补给、补给量、补给后是否回矿山等一系列的问题。在这一关中，我们考虑使用模拟的手段，以期得到一个最优策略，使得玩家最终的资金数达到最大。

6.2 “第三关”模型的建立

由于挖矿的基础收益比较少（只有 200 元），经过计算后，我们发现在高温天气坚持挖矿（花费 405 元）是很亏本的，为了使最后保留的资金最多，我们将高温天气挖矿这一选择去掉，即若玩家在矿场出现高温天气，玩家只能选择在原地等待之后的晴天或者直接走到下一个位置。除此之外，因为玩家到达终点后退回剩余的水和食物是以每箱为基准价格的一半作为退还价格，玩家在起点位置购买地资源不仅要能支撑玩家走到终点，还要尽可能少地购买资源以避免更多地损失。

首先，我们需要将“第三关”中的地图抽象成一个有向图 (如图 5)。有图可知，因

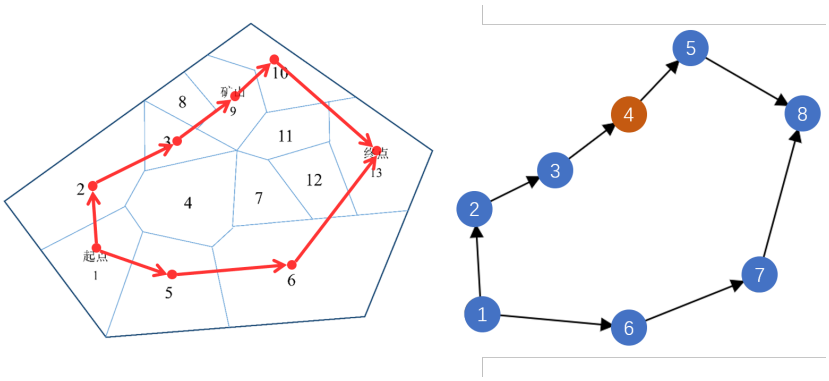

图 5　“第三关”有向图

为直接去终点的最短路与到矿山再到终点的路不重合，所以玩家需要在起点位置确定自己的基本方针——去矿山挖矿或者直接走向终点。

接着，我们根据这个有向图，制定了四种策略并进行模拟。

6.2.1 策略一

对于策略一，我们做以下规定：

(1) 玩家在起点位置决定去矿山挖矿。在最初的三天（从起点到达矿山）和最后的两天（从矿山到达终点），高温天停留在原地，晴天继续走。

(2) 玩家达到矿山后，只在晴天下挖矿，在高温下选择停留。

(3) 玩家在起点位置购买的资源为在路上的五天消耗的资源（以全高温天计）加上剩下的天数在晴天下在矿山挖矿的消耗的资源数。

$$F_{initial} = (d_{mine} + d_{dest}) * f_{high} * 2 + (d_{period} - (d_{mine} + d_{dest})) * f_{sun} * 3$$

$$W_{initial} = (d_{mine} + d_{dest}) * w_{high} * 2 + (d_{period} - (d_{mine} + d_{dest})) * w_{sun} * 3$$

因为高温天只矿山停留，水和食物的基础消耗都为 9 箱，而晴天挖矿的水消耗为 9 箱，食物为 12 箱，因此，在购买资源的时候，在矿山位置上消耗的所有资源都可以以晴天挖矿情形进行计算，这样就可以保证玩家在游戏过程中不会因资源消耗完毕而失败，也可以尽可能少地购买资源。

6.2.2 策略二

对于策略二，我们做以下规定：

(1) 玩家在起点位置决定去矿山挖矿。在最初的三天（从起点到达矿山）和最后的两天（从矿山到达终点），只能选择走到下一个位置。

(2) 玩家达到矿山后，只在晴天下挖矿，在高温下选择停留。

(3) 玩家在起点位置购买的资源为在路上的五天消耗的资源（以全高温天计）加上剩下的天数在晴天下在矿山挖矿的消耗的资源数。

$$F_{initial} = (d_{mine} + d_{dest}) * f_{high} * 2 + (d_{period} - (d_{mine} + d_{dest})) * f_{sun} * 3$$

$$W_{initial} = (d_{mine} + d_{dest}) * w_{high} * 2 + (d_{period} - (d_{mine} + d_{dest})) * w_{sun} * 3$$

因为高温天只矿山停留，水和食物的基础消耗都为 9 箱，而晴天挖矿的水消耗为 9 箱，食物为 12 箱，因此，在购买资源的时候，在矿山位置上消耗的所有资源都可以以晴天挖矿情形进行计算，这样就可以保证玩家在游戏过程中不会因资源消耗完毕而失败，也可以尽可能少地购买资源。

6.2.3 策略三

对于策略三，我们做以下规定：

(1) 玩家在起点位置决定走直达终点的最短路径，直接到达终点。并且在路中，出现高温天气停留在原地，只有在晴天天气下才走路。

(2) 玩家在起点位置购买的资源为在路上的行走消耗的资源（以全高温天计）加上剩下的天数在高温天下在路上停留消耗的资源数。

$$F_{initial} = d_{mine} * f_{high} * 2 + (d_{period} - d_{mine}) * f_{high}$$

$$W_{initial} = d_{mine} * w_{high} * 2 + (d_{period} - d_{mine}) * w_{high}$$

直接去终点走路总共只需要 3 天时间，且在高温天气停留在原地，晴天走路，以这种方式购买资源就可以保证玩家在游戏过程中不会因资源消耗完毕而失败，也可以尽可能少地购买资源。

6.2.4 策略四

对于策略四，我们做以下规定：

(1) 玩家在起点位置决定走直达终点的最短路径，直接到达终点。并且，不计天气状况，一直走，直到到达终点，总计 3 天。

(2) 玩家在起点位置购买的资源为在路上的行走消耗的资源（以全高温天计）。

$$F_{initial} = d_{mine} * f_{high} * 2$$

$$W_{initial} = d_{mine} * w_{high} * 2$$

直接去终点走路总共只需要 3 天时间，那我们需要考虑最差的情形——3 天都为高温天气。以这种方式购买资源就可以保证玩家在游戏过程中不会因资源消耗完毕而失败，也可以尽可能少地购买资源。

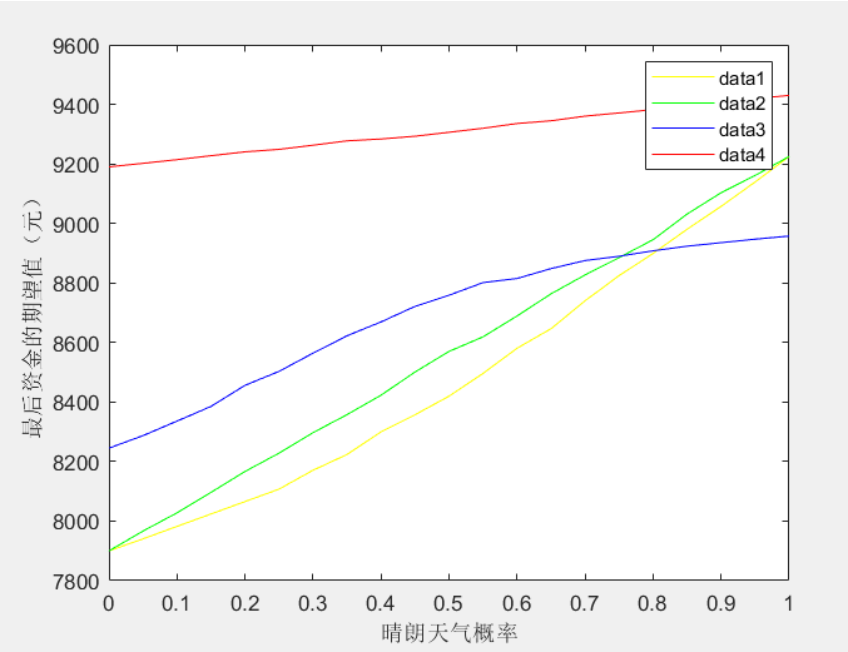

图 **6**　四种策略比较

6.3 “关卡三”模型的求解

经过 matlab 模拟，我们发现不论天气直接花三天走到终点，最终得到的资金最多，最终资金在 $[9190, 9430]$ 元之间波动，而其他情形的资金都小于这种情况。因此我们采用策略四，即不论天气如何直接向终点走，走三天到达终点。

6.4 “关卡四”模型的建立

根据前面的分析，玩家需要尽可能多地待在矿山，如果需要去村庄补给且剩余天数较多，否则，直接返回终点。因此，在 matlab 模拟过程中，我们给出了以下限制条件：

(1) 出现晴朗天气的概率为 p_{sun}。由于沙暴天气较少出现，设定其出现概率为 $p_{storm} = 0.05$, 且不能连续两天都为沙暴天气。

(2) 玩家从起点选择最短路到达矿山。到达矿山后，就一直挖矿，资源不足时考虑是否去村庄补给。

(3) 玩家身处矿山位置时，如果距离游戏结束的天数大于 d_{rest1} 天且玩家目前所有的资源，大于当天挖矿所消耗的资源与两天高温天气行走的消耗的资源以及一天因沙暴停留在路上所消耗的资源的总和，则当天继续在矿山挖矿；如果距离游戏结束的天数大于 d_{rest1} 天但资源没有达到上述要求，需要到村庄补给。

即，若

$$d_{period} - d_{pass} > d_{rest1}$$

且

$$F_{initial} - f > 3 * wc(W_i) + 2 * 2 * f_{high} + f_{storm}$$

则继续挖矿；

若

$$d_{period} - d_{pass} > d_{rest1}$$

且

$$F_{initial} - f \leq 3 * wc(W_i) + 2 * 2 * f_{high} + f_{storm}$$

则去村庄补给。

(4) 玩家在村庄补给时，如果距离游戏结束的天数小于 d_{rest2}，直接回终点。如果距离游戏结束的天数大于 d_{rest2}，需要考虑回矿山。除此之外，购买的资源也要考虑到上述天数，且资源要按照之前总的水的消耗与食物的消耗的比例进补充。

即，若

$$d_{period} - d_{pass} > d_{rest2}$$

则回矿山；

若

$$d_{period} - d_{pass} \leq d_{rest2}$$

则直接去终点。

6.5 ”关卡四”模型的求解

在模拟过程中，我们发现，当全部为晴天天气时且不考虑去村庄补给，我们最后的资金为 20162.5 元；对于晴天天气的其他概率，我们多次抽样得到钱数取平均值。不断调整参数后，我们发现 $d_{rest1} = 10$,$d_{rest} = 5$ 时最后的资金较多。然后，调整起始位置购买的资源的数量我们发现以下情况能达到最大资金数期望为 18080 元。

七、 问题三的模型建立和求解

7.1 问题三分析

问题三中是多名玩家一起进行游戏，且初始条件完全相同，但两个人的行动受到彼此的一些限制，因此我们需要综合考量这些玩家的行动方案。由于这些限制的存在，玩家们在游戏过程要想最终取得最多的资金数，需要彼此之间互相合作。

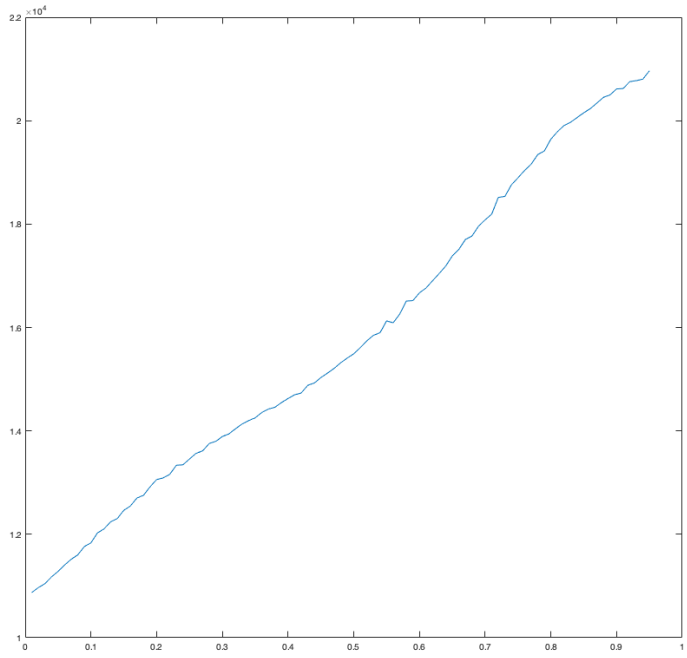

图 7　随着概率变化的收益变化图

7.1.1 “第五关”分析

“第五关”中，总共的游戏时间为十天，两名玩家需要在给定游戏时间内到达终点且资源在游戏过程中不能出现完全被消耗的情形。由于地图中没有村庄，两名玩家需要在起点位置购买足够的水和食物，且不能用完初始资金也不能超过负重上限。我们考虑使用模拟的手段，使得两名玩家最终的资金总数之和最大。

7.1.2 “第六关”分析

“第六关”中，总共的游戏时间为十天，三名玩家需要在给定游戏时间内到达终点且资源在游戏过程中不能出现完全被消耗的情形。由于玩家只知道当天的天气情况，所以我们要给晴朗天气、沙暴天气等的出现设定一个概率。除此之外，要考虑到三名玩家的相互制约条件，我们采用模拟的方法，使得三名玩家最终的资金总数之和最大。

7.2 “第五关”模型的建立

由于“第五关”的游戏设定与“第三关”的游戏设定基本一致，只是从一名玩家变成了两名玩家，因此需要设定两者关系为合作关系——即游戏最后使得两者资金总和达到最大。

考虑到“第三关”的结果——因为矿山的每日基础收益比较少，玩家应该直接从起

点到达终点。那么“第五关”的两名玩家也应该尽量减少前往矿山挖矿的情形，采取合作的方法，以双方收益总和作为判别标准。因此，基于合作博弈的思想，我们给出以下两种策略。

7.2.1 策略一

因为两名玩家如果在同一天从区域 A 到区域 B（A、B 不相同），则消耗翻倍，所以两名玩家在本关卡只走两条不相交的路线。除此之外，根据计算，去矿山挖矿的收益过低而路上消耗过高，因此两名玩家都不去矿山挖矿。除此之外，由于高温天气走路消耗过多，两名玩家都需要在第二天（高温天气）停留一天，等到晴朗天气再走到下一个位置。（路线图见图 8）

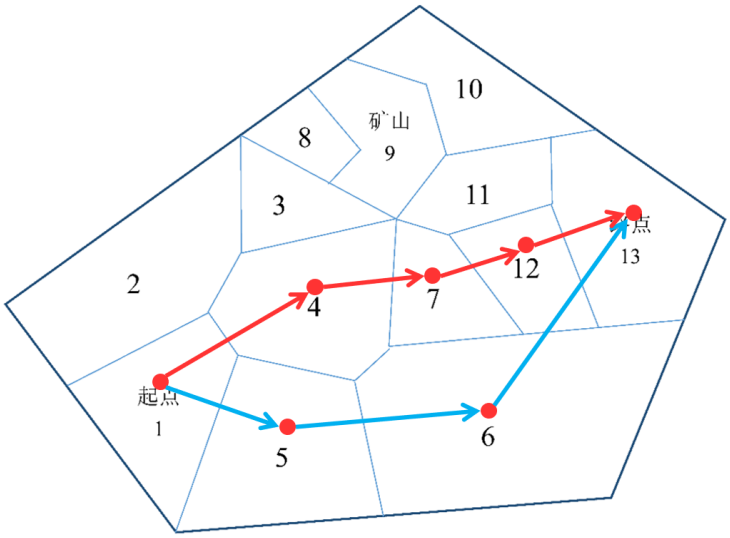

图 **8**　“第五关”策略一

7.2.2 策略二

因为两名玩家如果在同一天从区域 A 到区域 B（A,B 不相同）则消耗翻倍，所以两名玩家在本关卡选择错开位置的路线。除此之外，根据计算，去矿山挖矿的收益过低而路上消耗过高，因此两名玩家都不去矿山挖矿。除此之外，虽然高温天气消耗比较多，但为了少在路上走一天，一名玩家需要在第二天（高温天气）行走，而另一名玩家在原地停留一天。（路线图见图 9）

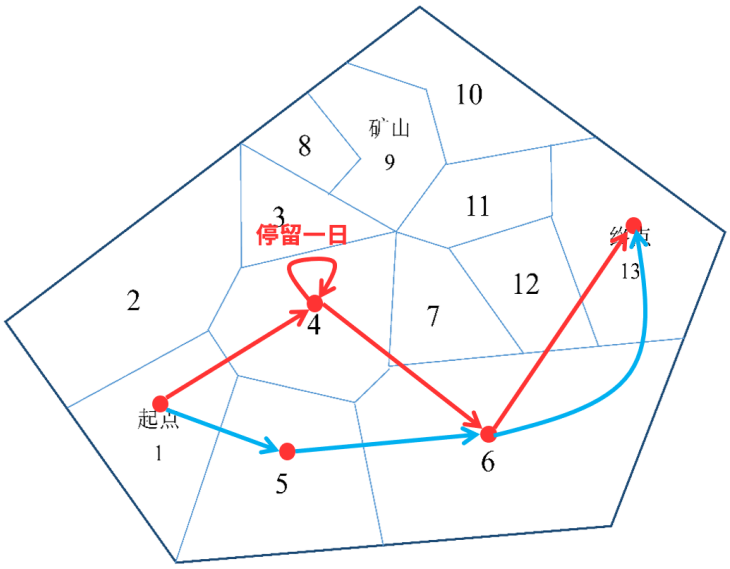

图 **9**　“第五关”策略二

7.3 “第五关”模型的求解

根据计算，策略一的两名玩家最后分别剩下了 9630 元和 9535 元，总计 19165 元；策略二的两名玩家最后分别剩下了 9645 元和 9535 元，总计 19180 元。因此策略二优于策略一，我们选择策略二——两名玩家都直接从起点到达终点，且为了错开位置，一名玩家在第二天在位置 4 停留一天再经过两天走到终点，一名玩家从起点直接经过三天到达终点。

7.4 “第六关”模型的建立

“第六关”的设定与“第四关”的设定基本一致，只是玩家数由一人变成了三人（三名玩家分别为 A、B、C）。根据“第四关”的策略，我们需要考虑到以下因素：一、在非矿山位置每天有两种选择——停留和走一步，在矿山位置每天有三种选择——停留、走一步和挖矿。二、由于该地图中矿山的收益比较高（挖矿的每天基础收益为 1000 元），要想达到最大收益，玩家需要尽可能多地待在矿山挖矿。三、由于有村庄的存在，玩家需要考量起始位置带的资源数、是否前去村庄补给、补给量、补给后是否回矿山等一系列的问题。在这一关中，我们同样使用模拟的手段，将三名玩家设定为合作关系，最终的目标为游戏最后使得两者资金总和达到最大。

7.4.1 策略

(1) A 和 B 两名玩家从起点处出发选择两条不同的路线到达矿山，玩家 C 先在起点等候一天，再出发。这样可以避免三个人中的其中两个同时从区域 A 到区域 B 这样情况的发生。（如图 10）

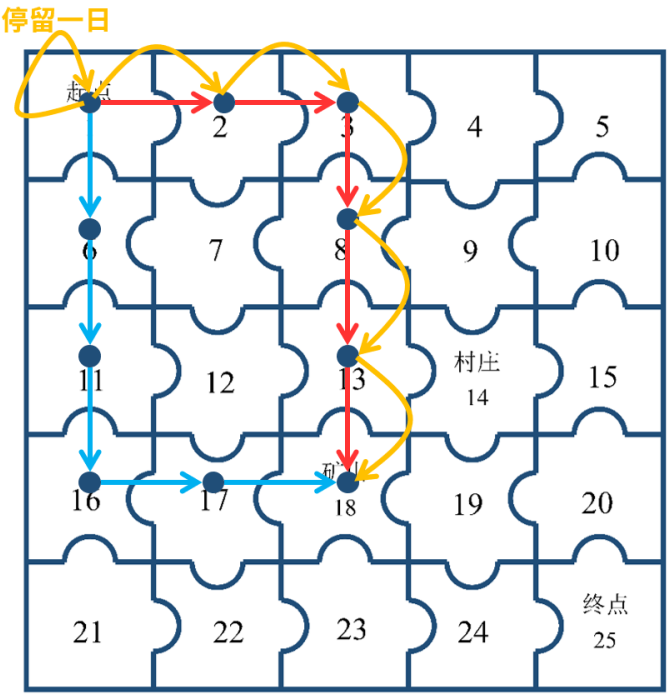

图 **10**　“第六关”三名玩家从起点到矿山的路线图

(2) A,B,C 三名玩家到达矿山后，每天有一人挖矿，其他两个人在原地停留。首先是玩家 A 挖矿，一直挖到资源不够；玩家 B 开始挖；以此类推直到玩家的 C 的资源不够。

(3) 每名玩家挖完后都需要判断是否去村庄补给，补给多少资源以及补给完是否有必要回到矿山等待下一轮挖矿。

7.5 ”第六关” 模型的求解

经过计算机模拟后，我们发现 A,B,C 三名玩家到达矿山后，每天有一人挖矿，其他两个人在原地停留。首先是玩家 A 挖矿，一直挖到资源不够；玩家 B 开始挖；以此类推直到玩家的 C 的资源不够。设定晴朗天气出现的概率为 0.5，沙暴天气出现的概率为 0.05。第一名玩家挖矿完，游戏时间剩余 14 天；第二名玩家挖矿完，游戏时间剩余 12 天；因为第三个人挖完后只剩下 8 天时间，所以每个人挖完后，不需要去村庄进行补给，可以直接回终点。除此之外，当在初始位置购买的水和食物的箱数配比为 233:250 时，游戏结束时的资金数可达 32522.89 元。

八、 模型的优缺点

8.1 模型的优点

1. 使用了动态规划的方法，能够准确地得到最优解。

2. 进行了大量多次的仿真模拟。

8.2 模型的缺点

1. 动态规划程序运行较慢，若时间充裕，可以进行优化。

2. 由于时间紧张，只考虑一些主要的策略；若时间充裕，可以多考虑一些情况并且多做几次模拟。

参考文献

[1]《运筹学》教材编写组. 《运筹学》. 第 4 版. 清华大学出版社.

[2] 谢识予. 《经济学博弈论》. 第 4 版. 复旦大学出版社.

附录 A　第一问动态规划

```
#include <iostream>

using namespace std;

int prof[31][15][431][501];
int WC[4]={0, 5, 8, 10};
int FC[4]={0, 7, 6, 10};
int Weath[31];
int pos[30][6];
int str[31][15][431][501][3];

int main()
{
for ( int i=1 ; i<=30 ; i++)
cin>>Weath[i];
cout<<"OK!"<<endl;
int maxpos;
cin>>maxpos;
cout<<maxpos<<endl;
for ( int i=1; i<=maxpos; i++)
{
cin>>pos[i][0]>>pos[i][1];
for ( int j=1; j<=pos[i][1]; j++)
cin>>pos[i][j+1];
}
cout<<11<<endl;
cout<<pos[11][0]<<' '<<pos[11][1]<<' '<<pos[11][2]<<' '<<pos[11][3]<<endl;

cout<<"OK!"<<endl;

int T, P, W, F;
for(T=0; T<=30; T++)
for ( P=1 ; P<=maxpos ; P++)
for ( W=0; W<=430; W++)
for ( F=0; F<=500; F++)
prof[T][P][W][F]=-1000000;
for ( W=0; W<=400; W++)
for (F=0; F<=(1200-3*W)/2&&F<=450; F++)
prof[0][1][W][F]=10000-5*W-10*F;

int maxvalue=0;
int maxday=-1;

for (T=1; T<=30; T++)
```

```
{
cout<<T<<endl;
if (Weath[T]!=3)   //不是风暴
{
int tmpmax=-1;
for (int i=1; i<=pos[maxpos][1]; i++)
{
if (prof[T-1][pos[maxpos][i+1]][2*WC[Weath[T]]][2*FC[Weath[T]]]>tmpmax)
{
tmpmax=prof[T-1][pos[maxpos][i+1]][2*WC[Weath[T]]][2*FC[Weath[T]]];
str[T][maxpos][0][0][0]=pos[maxpos][i+1];
str[T][maxpos][0][0][1]=2*WC[Weath[T]];
str[T][maxpos][0][0][2]=2*FC[Weath[T]];
}
}
if (tmpmax>maxvalue)
{
maxvalue=tmpmax;
maxday=T;
cout<<tmpmax<<endl;
}
if (T==30)
break;
for ( P=1; P<maxpos; P++)
for ( W=0; W<=400; W++)
for (F=0; F<=(1200-3*W)/2 && F<=450; F++)
{
if (pos[P][0]==0 ) //一般地带
{
prof[T][P][W][F]=prof[T-1][P][W+WC[Weath[T]]][F+FC[Weath[T]]]; //原地不动
str[T][P][W][F][0]=P;
str[T][P][W][F][1]=W+WC[Weath[T]];
str[T][P][W][F][2]=F+FC[Weath[T]];
for (int i=1; i<=pos[P][1]; i++)
{
if (prof[T-1][pos[P][i+1]][W+2*WC[Weath[T]]][F+2*FC[Weath[T]]]>prof[T][P][W][F])
{
prof[T][P][W][F]=prof[T-1][pos[P][i+1]][W+2*WC[Weath[T]]][F+2*FC[Weath[T]]];
str[T][P][W][F][0]=pos[P][i+1];
str[T][P][W][F][1]=W+2*WC[Weath[T]];
str[T][P][W][F][2]=F+2*FC[Weath[T]];
}
}
}
if (pos[P][0]==1 ) //矿山
{
prof[T][P][W][F]=prof[T-1][P][W+WC[Weath[T]]][F+FC[Weath[T]]]; //原地不动
```

```
str[T][P][W][F][0]=P;
str[T][P][W][F][1]=W+WC[Weath[T]];
str[T][P][W][F][2]=F+FC[Weath[T]];
if (prof[T-1][P][W+3*WC[Weath[T]]][F+3*FC[Weath[T]]]+1000>prof[T][P][W][F]) //挖矿
{
prof[T][P][W][F]=prof[T-1][P][W+3*WC[Weath[T]]][F+3*FC[Weath[T]]]+1000;
str[T][P][W][F][0]=P;
str[T][P][W][F][1]=W+3*WC[Weath[T]];
str[T][P][W][F][2]=F+3*FC[Weath[T]];
}
for (int i=1; i<=pos[P][1]; i++)
{
if (prof[T-1][pos[P][i+1]][W+2*WC[Weath[T]]][F+2*FC[Weath[T]]]>prof[T][P][W][F])
{
prof[T][P][W][F]=prof[T-1][pos[P][i+1]][W+2*WC[Weath[T]]][F+2*FC[Weath[T]]];
str[T][P][W][F][0]=pos[P][i+1];
str[T][P][W][F][1]=W+2*WC[Weath[T]];
str[T][P][W][F][2]=F+2*FC[Weath[T]];
}
}
}
if (pos[P][0]==2 ) //村庄
{
prof[T][P][W][F]=prof[T-1][P][W+WC[Weath[T]]][F+FC[Weath[T]]]; //原地不动
str[T][P][W][F][0]=P;
str[T][P][W][F][1]=W+WC[Weath[T]];
str[T][P][W][F][2]=F+FC[Weath[T]];
/*
for (int i=0; i<=W ; i++)
for (int j=0; j<=F; j++)
{
if (prof[T-1][P][W+WC[Weath[T]]-i][F+FC[Weath[T]]-j]-10*i-20*j>prof[T][P][W][F])
{
prof[T][P][W][F]=prof[T-1][P][W+WC[Weath[T]]-i][F+FC[Weath[T]]-j]-10*i-20*j;
str[T][P][W][F][0]=P;
str[T][P][W][F][1]=W+WC[Weath[T]]-i;
str[T][P][W][F][2]=F+FC[Weath[T]]-j;
}
}
*/
for (int k=1; k<=pos[P][1]; k++)
for (int i=0; i<=W ; i++)
for (int j=0; j<=F; j++)
{
if (prof[T-1][pos[P][k+1]][W+2*WC[Weath[T]]-i][F+2*FC[Weath[T]]-j]-10*i-20*j>prof[T][P][W][F])
{
prof[T][P][W][F]=prof[T-1][pos[P][k+1]][W+2*WC[Weath[T]]-i][F+2*FC[Weath[T]]-j]-10*i-20*j;
```

```
str[T][P][W][F][0]=pos[P][k+1];
str[T][P][W][F][1]=W+2*WC[Weath[T]]-i;
str[T][P][W][F][2]=F+2*FC[Weath[T]]-j;
}
}
}
}
}
if (Weath[T]==3)   //是风暴
{
for ( P=1; P<maxpos; P++)
for ( W=0; W<=400; W++)
for (F=0; F<=(1200-3*W)/2 && F<=450; F++)
{
if (pos[P][0]==0 ) //一般地带
{
prof[T][P][W][F]=prof[T-1][P][W+WC[Weath[T]]][F+FC[Weath[T]]]; //原地不动
str[T][P][W][F][0]=P;
str[T][P][W][F][1]=W+WC[Weath[T]];
str[T][P][W][F][2]=F+FC[Weath[T]];

}
if (pos[P][0]==1 ) //矿山
{
prof[T][P][W][F]=prof[T-1][P][W+WC[Weath[T]]][F+FC[Weath[T]]]; //原地不动
str[T][P][W][F][0]=P;
str[T][P][W][F][1]=W+WC[Weath[T]];
str[T][P][W][F][2]=F+FC[Weath[T]];
if (prof[T-1][P][W+3*WC[Weath[T]]][F+3*FC[Weath[T]]]+1000>prof[T][P][W][F]) //挖矿
{
prof[T][P][W][F]=prof[T-1][P][W+3*WC[Weath[T]]][F+3*FC[Weath[T]]]+1000;
str[T][P][W][F][0]=P;
str[T][P][W][F][1]=W+3*WC[Weath[T]];
str[T][P][W][F][2]=F+3*FC[Weath[T]];
}
}
if (pos[P][0]==2 ) //村庄
{
prof[T][P][W][F]=prof[T-1][P][W+WC[Weath[T]]][F+FC[Weath[T]]]; //原地不动
str[T][P][W][F][0]=P;
str[T][P][W][F][1]=W+WC[Weath[T]];
str[T][P][W][F][2]=F+FC[Weath[T]];
for (int i=0; i<=W ; i++)
for (int j=0; j<=F; j++)
{
if (prof[T-1][P][W+WC[Weath[T]]-i][F+FC[Weath[T]]-j]-10*i-20*j>prof[T][P][W][F])
{
```

```
prof[T][P][W][F]=prof[T-1][P][W+WC[Weath[T]]-i][F+FC[Weath[T]]-j]-10*i-20*j;
str[T][P][W][F][0]=P;
str[T][P][W][F][1]=W+WC[Weath[T]]-i;
str[T][P][W][F][2]=F+FC[Weath[T]]-j;
}
}
}
}
}
}
cout<<maxvalue<<' '<<maxday<<endl;
cout<<maxday<<' '<<maxpos<<' '<<0<<' '<<0<<' '<<maxvalue<<endl;
int tmpP=str[maxday][maxpos][0][0][0];
int tmpW=str[maxday][maxpos][0][0][1];
int tmpF=str[maxday][maxpos][0][0][2];
for (int i=maxday-1; i>=0 ; i--)
{
cout<<i<<' '<<tmpP<<' '<<tmpW<<' '<<tmpF<<' '<<prof[i][tmpP][tmpW][tmpF]<<endl;
int a=str[i][tmpP][tmpW][tmpF][0];
int b=str[i][tmpP][tmpW][tmpF][1];
int c=str[i][tmpP][tmpW][tmpF][2];
tmpP=a;
tmpW=b;
tmpF=c;
}
return 0;
}
```

附录 B “第三关”最优策略模拟代码

```
iteration = 1000;
time = 10;
action_mat = zeros(iteration,time);

p = [0:0.05:1]; % p \in [0,1]
m = size(p,2);
income_mean = zeros(m,1);
for k = 1:m
period = 10; % period \in [5,10]
income_mat = zeros(iteration,1);
for i = 1:iteration
[actmat,income] = sol_4(weather(p(k),time),time,period);
actmat = actmat';
action_mat(i,:) = actmat(:,:);
income_mat(i) = income;
```

```
end
income_mean(k) = mean(income_mat);
end

plot(p,income_mean,'k');
hold on

function [weamat] = weather(prob,time)
%Generating a weather distribution matrix, prob is the possibility of sunny
%days, time is the length of the period, return 1 is sunny, 0 high temp
weamat = zeros(time,1);
for i = 1:time
if rand() < prob
weamat(i) = 1;
end
end

end



function [action,income] = sol_1(weamat,time,period)
%water and food, units of bringing, minday minimun day to go back, period
%is the total days within the dessert
dist_mine = 3;
dist_dest = 2;
minday = dist_dest + dist_mine;
action = zeros(time,1);
for i = 1:time
action(i) = -1;
end
income = 10000;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
food = minday * hitemp_unit_food * walkrat+(period-minday)*craftrat*sunny_unit_food;
water = minday * hitemp_unit_water * walkrat+(period-minday)*craftrat*sunny_unit_water;
food_cost = food*food_price;
water_cost = water*water_price;
income = income - food_cost - water_cost;
walk_count = 0;
restday = period - minday;
for i = 1:period
```

```
if weamat(i) == 0 && restday > 0
action(i) = 0;
restday = restday - 1;
food = food - hitemp_unit_food;
water = water - hitemp_unit_water;
else
action(i) = 1;
food = food - (1 - weamat(i))*walkrat*hitemp_unit_food-weamat(i)*walkrat*sunny_unit_food;
water = water - (1 - weamat(i))*walkrat*hitemp_unit_water-weamat(i)*walkrat*sunny_unit_water;
walk_count = walk_count + 1;
if walk_count == dist_mine
break;
end
end
end

for j = i+1:i+restday
if weamat(j) == 1
action(j) = 2;
food = food - sunny_unit_food * craftrat;
water = water - sunny_unit_water * craftrat;
income = income + 200;
else
action(j) = 0;
food = food - hitemp_unit_food;
water = water - hitemp_unit_water;
end
end

j = i + restday;

food = food - walkrat*(weamat(j+1)+weamat(j+2))*sunny_unit_food - (2 - weamat(j+1) -
    weamat(j+2))*hitemp_unit_food*walkrat;
water = water - walkrat*(weamat(j+1)+weamat(j+2))*sunny_unit_water - (2 - weamat(j+1) -
    weamat(j+2))*hitemp_unit_water*walkrat;
action(j+1) = 1;
action(j+2) = 1;
walk_count = walk_count + 2;
income = income + food*food_price/2 + water*water_price/2;
end


function [action,income] = sol_2(weamat,time,period)
%water and food, units of bringing, minday minimun day to go back, period
%is the total days within the dessert
dist_mine = 3;
dist_dest = 2;
```

```
minday = dist_dest + dist_mine;
action = zeros(time,1);
for i = 1:time
action(i) = -1;
end
income = 10000;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
food = minday * hitemp_unit_food * walkrat+(period-minday)*craftrat*sunny_unit_food;
water = minday * hitemp_unit_water * walkrat+(period-minday)*craftrat*sunny_unit_water;
food_cost = food*food_price;
water_cost = water*water_price;
income = income - food_cost - water_cost;
walk_count = 0;
restday = period - minday;
for i = 1:dist_mine
action(i) = 1;
food = food - (1 - weamat(i))*walkrat*hitemp_unit_food-weamat(i)*walkrat*sunny_unit_food;
water = water - (1 - weamat(i))*walkrat*hitemp_unit_water-weamat(i)*walkrat*sunny_unit_water;
walk_count = walk_count + 1;
end

for j = i+1:i+restday
if weamat(j) == 1
action(j) = 2;
food = food - sunny_unit_food * craftrat;
water = water - sunny_unit_water * craftrat;
income = income + 200;
else
action(j) = 0;
food = food - hitemp_unit_food;
water = water - hitemp_unit_water;
end
end

j = i + restday;

food = food - walkrat*(weamat(j+1)+weamat(j+2))*sunny_unit_food - (2 - weamat(j+1) -
    weamat(j+2))*hitemp_unit_food*walkrat;
water = water - walkrat*(weamat(j+1)+weamat(j+2))*sunny_unit_water - (2 - weamat(j+1) -
    weamat(j+2))*hitemp_unit_water*walkrat;
```

```
action(j+1) = 1;
action(j+2) = 1;
walk_count = walk_count + 2;
income = income + food*food_price/2 + water*water_price/2;
end

function [action,income] = sol_3(weamat,time,period)
%water and food, units of bringing, minday minimun day to go back, period
%is the total days within the dessert
minday = 3;
action = zeros(time,1);
for i = 1:time
action(i) = -1;
end
income = 10000;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
walkrat = 2;
food_price = 10;
water_price = 5;
food = minday * hitemp_unit_food * walkrat+(period-minday)*hitemp_unit_food;
water = minday * hitemp_unit_water * walkrat+(period-minday)*hitemp_unit_water;
food_cost = food*food_price;
water_cost = water*water_price;
income = income - food_cost - water_cost;
walk_count = 0;
restday = period - minday;
for i = 1:period
if weamat(i) == 0 && restday > 0
action(i) = 0;
restday = restday - 1;
food = food - hitemp_unit_food;
water = water - hitemp_unit_water;
else
action(i) = 1;
food = food - (1 - weamat(i))*walkrat*hitemp_unit_food-weamat(i)*walkrat*sunny_unit_food;
water = water - (1 - weamat(i))*walkrat*hitemp_unit_water-weamat(i)*walkrat*sunny_unit_water;
walk_count = walk_count + 1;
if walk_count == minday
break;
end
end
end

income = income + food*food_price/2 + water*water_price/2;
```

```
end


function [action,income] = sol_4(weamat,time,period)
%water and food, units of bringing, minday minimun day to go back, period
%is the total days within the dessert
minday = 3;
action = zeros(time,1);
for i = 1:time
action(i) = -1;
end
income = 10000;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
walkrat = 2;
food_price = 10;
water_price = 5;
food = minday * hitemp_unit_food * walkrat;
water = minday * hitemp_unit_water * walkrat;
food_cost = food*food_price;
water_cost = water*water_price;
income = income - food_cost - water_cost;
for i = 1:minday
action(i) = 1;
food = food - (1 - weamat(i))*walkrat*hitemp_unit_food-weamat(i)*walkrat*sunny_unit_food;
water = water - (1 - weamat(i))*walkrat*hitemp_unit_water-weamat(i)*walkrat*sunny_unit_water;
end

income = income + food*food_price/2 + water*water_price/2;
end
```

附录 C “第四关”最优策略模拟代码

```
initwat = 144+(304-144)*rand(1,10000);%inital condition of water and food, respectively
initfood = (1200-initwat.*3)./2;
p1 = [0.3;0.05;0.7];
for k = 1:1000
for i=1:size(p1,2)
%    sizep = size(p1,2);
%    income2 = zeros(sizep,1);
%    p2 = 0.05;% possibility of storm
%    for i = 1:sizep
[action,income] = sol_a(weather(p1(i),p2,30),initwat(k),initfood(k),30,30);
```

```
income1(i) = income;
%    end
end
incomemean(k) = mean(income1);
end
[x y]=find(incomemean==max(max(incomemean)))
max(max(incomemean))
initwat(y)
initfood(y)


function [weamat] = weather(prob1,prob2,time)
%Generating a weather distribution matrix, prob is the possibility of sunny
%days, time is the length of the period, return 1 is sunny, 0 high temp
weamat = zeros(time,1);
for i = 1:time
a = rand();
if a < prob1
weamat(i) = 1;
elseif a >= prob1 && a < prob1 + prob2
weamat(i) = 2;
else
weamat(i) = 0;
end
end

end


function [water,food] = costfun(water,food,current_wea,status)
%status- 0 stand, 1 walk, 2 mining, weather - 0 high temp, 1 sunny, 2 storm
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;

if current_wea == 0
```

```
watrat = hitemp_unit_water;
foodrat = hitemp_unit_food;
elseif current_wea == 1
watrat = sunny_unit_water;
foodrat = sunny_unit_food;
else
watrat = storm_unit_water;
foodrat = storm_unit_food;
end

if status == 0
costrat = 1;
elseif status == 1
costrat = walkrat;
else
costrat = craftrat;
end

food = food - costrat * foodrat;
water = water - costrat * watrat;

end



function [food,water,action,index_day,income] =
    decision_making(food,water,weamat,action,restday,index_day,income,period,initwat,initfood)
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
flag = -1;
for j = index_day+1:period
decision = mine_storage_check(food,water,weamat(j),restday);
if j < flag
continue;
elseif decision == 0
```

```
[food,water,action,j] = walking(food,water,dist_mine_vill,j,weamat,action,period);
restday = period - j;
[income,food,water,restday,action,j,dest_id] =
    vill_decision(income,food,water,restday,weamat,j,action,period,initwat,initfood);
flag = j+1;
if dest_id == 1;
flag = -1;
break;
end
elseif decision == 2
[food,water,action,j] = walking(food,water,dist_dest,j,weamat,action,period);
restday = 0;
break;
elseif decision == 1
[water,food] = costfun(water,food,weamat(j),2);
action(j) = 2;
restday = period - j;
income = income + 1000;
end
end
index_day = j;
end


function [food,water,action,index_day] =
    walking(food,water,steps,index_day,weamat,action,period)
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
walk_count = 0;
for i = index_day:period
if weamat(i) == 0
food = food - walkrat*hitemp_unit_food;
water = water - walkrat*hitemp_unit_water;
walk_count = walk_count + 1;
action(i) = 1;
```

```
elseif weamat(i) == 1
food = food - walkrat*sunny_unit_food;
water = water - walkrat*sunny_unit_water;
walk_count = walk_count + 1;
action(i) = 1;
else
food = food - storm_unit_food;
water = water - storm_unit_water;
action(i) = 0;
end
if walk_count == steps
break;
end
end
index_day = i;

end

function [id] = mine_storage_check(food,water,current_wea,restday)
%return 0 if should go to village, 2 if should directly to destination, 1
%means continue mining, restday including the day input
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
if current_wea == 0
watrat = hitemp_unit_water;
foodrat = hitemp_unit_food;
elseif current_wea == 1
watrat = sunny_unit_water;
foodrat = sunny_unit_food;
else
watrat = storm_unit_water;
foodrat = storm_unit_food;
end
remain_food = food - craftrat*foodrat;
remain_water = water - craftrat*watrat;
```

```
if restday > 10%value may be modified
if remain_food >= (2 * hitemp_unit_food * walkrat + storm_unit_food )&& remain_water >=
    (2*hitemp_unit_water*walkrat + storm_unit_water)
id = 1;
else
id = 0;
end
elseif restday > 4
if remain_food >= (3 * hitemp_unit_food * walkrat + storm_unit_food) && remain_water >=
    (3*hitemp_unit_water*walkrat + storm_unit_water)
id = 1;
else
id = 2;
end
else
id = 2;
end

end

function [action,income] = sol_a(weamat,water,food,time,period)
%water and food, units of bringing, minday minimun day to go back, period
%is the total days within the dessert
initwat = water;
initfood = food;
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
action = zeros(time,1);
for i = 1:time
action(i) = -1;
end
income = 10000;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
food_cost = food*food_price;
water_cost = water*water_price;
income = income - food_cost - water_cost;
```

```
walk_count = 0;
for i = 1:period
if weamat(i) == 0
food = food - walkrat*hitemp_unit_food;
water = water - walkrat*hitemp_unit_water;
walk_count = walk_count + 1;
action(i) = 1;
elseif weamat(i) == 1
food = food - walkrat*sunny_unit_food;
water = water - walkrat*sunny_unit_water;
walk_count = walk_count + 1;
action(i) = 1;
else
food = food - storm_unit_food;
water = water - storm_unit_water;
action(i) = 0;
end
if walk_count == dist_mine
break;
end
end
restday = period - i;
index_day = i;
[food,water,action,index_day,income] =
    decision_making(food,water,weamat,action,restday,index_day,income,period,initwat,initfood);
index_day
income = income + food*food_price/2 + water*water_price/2;
end


function [income,food,water,restday,action,index_day,dest_id] =
    vill_decision(income,food,water,restday,weamat,index_day,action,period,initwat,initfood)
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
if restday > 5
```

```
back_id = 1;
[food,water,income] = addfw(food,water,income,restday,initwat,initfood,back_id);
index_day = index_day + 1;
[food,water,action,index_day] =
    walking(food,water,dist_mine_vill,index_day,weamat,action,period);
dest_id = 0;
else
back_id = 0;
[food,water,income] = addfw(food,water,income,restday,initwat,initfood,back_id);
index_day = index_day + 1;
[food,water,action,index_day] = walking(food,water,dist_dest,index_day,weamat,action,period);
dest_id = 1;
end
restday = period - index_day;
end

function [food,water,income] = addfw(food,water,income,restday,initwat,initfood,back_id)
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
food_weight = 2;
water_weight = 3;
fa = food;
wa = water;
if back_id == 1
[wa,fa] = costfun(wa,fa,0,1);
cost_food = (food - fa)*5 + (restday - 5)*craftrat*storm_unit_food;
cost_water = (water - wa)*5 + (restday - 5)*craftrat*storm_unit_water;
if (cost_food * food_weight + cost_water * water_weight) > 1200
costratio = (initfood - food) / (initwat - water);
food_final = costratio/(1+costratio) * 1200;
water_final = 1/(1+costratio)*1200;
food_final = floor(food_final / food_weight);
water_final = floor(water_final / water_weight);
income = income - (food_final - food)*food_price - (water_final - water)*water_price;
food = food_final;
```

```
water = water_final;
else
food_add = cost_food - food;
water_add = cost_water - water;
income = income - 2*food_add*food_price - 2*water_add*water_price;
food = cost_food;
water = cost_water;
end
else
[wa,fa] = costfun(wa,fa,0,1);
cost_food = (food - fa)*3 + storm_unit_food;
cost_water = (water - wa)*3 + storm_unit_water;
food_add = cost_food - food;
water_add = cost_water - water;
income = income - 2*food_add*food_price - 2*water_add*water_price;
food = cost_food;
water = cost_water;
end
end
```

附录 D “第六关”最优策略模拟代码

```
function [weamat] = weather(prob1,prob2,time)
%Generating a weather distribution matrix, prob is the possibility of sunny
%days, time is the length of the period, return 1 is sunny, 0 high temp
weamat = zeros(time,1);
for i = 1:time
a = rand();
if a < prob1
weamat(i) = 1;
elseif a >= prob1 && a < prob1 + prob2
weamat(i) = 2;
else
weamat(i) = 0;
end
end

end

function [action,initid,villid,villid2] =
    walking_decision(weamat,action,index_day,water,food,initid,mine_day)
%villid 2 to destination, 0 to village, 1 mining, 3 move one step, -1 stand
%  Detailed explanation goes here
period = 30;
restday = period - index_day;
```

```
villid = -1;
villid2 = -1;
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
rest_walking_day = max(0, dist_mine - index_day + 1);
restfood = food(1);
restwater = water(1);
metercount = walking_count(action,30,2);
metercount3 = walking_count(action,30,3);
remain_day = dist_mine - metercount ;
mining_count = remain_day - rest_walking_day;

for i = 1:rest_walking_day
if i == 1
[restwater,restfood] = costfun(restwater,restfood,weamat(index_day),1,1);
else
[restwater,restfood] = costfun(restwater,restfood,0,1,1);
end
end
for i = 1:mining_count
if rest_walking_day == 0
if i == 1
[restwater,restfood] = costfun(restwater,restfood,weamat(index_day),2,1);
else
[restwater,restfood] = costfun(restwater,restfood,2,2,1);
end
else
[restwater,restfood] = costfun(restwater,restfood,2,2,1);
end
end
for i = 1:3
[restwater,restfood] = costfun(restwater,restfood,0,1,1);
end

if remain_day > 0
```

```
if initid == 1
if weamat(index_day) == 0 && restwater >= 0 && restfood >= 0
action(2,index_day) = 0;
action(3,index_day) = 0;
elseif weamat(index_day) == 2
action(2,index_day) = 0;
action(3,index_day) = 0;
else
if index_day == 1
action(2,index_day) = 1;
villid = 3;
action(3,index_day) = 0;
initid = 0;
else
action(2,index_day) = 1;
action(3,index_day) = 1;
villid = 3;
villid2 = 3;
initid = 0;
end
end
else
if weamat(index_day) == 0 && restwater >= 0 && restfood >= 0
action(2,index_day) = 0;
action(3,index_day) = 0;
elseif weamat(index_day) == 2
action(2,index_day) = 0;
action(3,index_day) = 0;
else
action(2,index_day) = 1;
action(3,index_day) = 1;
villid = 3;
villid2 = 3;
end
end
elseif metercount3 < dist_mine
if action(1,index_day) == 2
action(2,index_day) = 0;
action(3,index_day) = 1;
villid2 = 3;
else
decision = mine_storage_check(food,water,weamat(index_day),restday,mine_day,2);
if decision == 1
action(2,index_day) = 2;
villid = 1;
else
villid = decision;
```

```
action(2,index_day) = 1;
end
if weamat(index_day) ~= 2
action(3,index_day) = 1;
villid2 = 3;
else
action(3,index_day) = 0;
end
end
else
if action(1,index_day) == 2
action(2,index_day) = 0;
action(3,index_day) = 1;
villid2 = 3;
else
decision1 = mine_storage_check(food,water,weamat(index_day),restday,mine_day,2);
if decision1 == 1
action(2,index_day) = 2;
villid = 1;
action(3,index_day) = 0;
else
villid = decision1;
action(2,index_day) = 1;
decision2 = mine_storage_check(food,water,weamat(index_day),restday,mine_day,3);
villid2 = decision2;
if decision2 == 1
action(3,index_day) = 2;
else
action(3,index_day) = 1;
end
end
end
end

end
function [counts] = walking_count(action,time,index)
%UNTITLED2 Summary of this function goes here
%  Detailed explanation goes here
counts = 1;
for i = 1: time
if action(index,i) == 1
counts = counts + 1;
end
end

end
```

```
function [food,water,action,index_day] =
    walking(food,water,steps,index_day,weamat,action,period,index)
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
walk_count = 0;
for i = index_day:period
if weamat(i) == 0
food(index) = food(index) - walkrat*hitemp_unit_food;
water(index) = water(index) - walkrat*hitemp_unit_water;
walk_count = walk_count + 1;
action(index,i) = 1;
elseif weamat(i) == 1
food(index) = food(index) - walkrat*sunny_unit_food;
water(index) = water(index) - walkrat*sunny_unit_water;
walk_count = walk_count + 1;
action(index,i) = 1;
else
food(index) = food(index) - storm_unit_food;
water(index) = water(index) - storm_unit_water;
action(index,i) = 0;
end
if walk_count == steps
break;
end
end
index_day = i;

end

function [income,food,water,restday,action,index_day,dest_id] =
    vill_decision(income,food,water,restday,weamat,index_day,action,period,initwat,initfood,vill_day,index)
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
```

```
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
if restday > vill_day %variable
back_id = 1;
[food,water,income] = addfw(food,water,income,restday,initwat,initfood,back_id,index);
index_day = index_day + 1;
[food,water,action,index_day] =
    walking(food,water,dist_mine_vill,index_day,weamat,action,period,index);
dest_id = 0;
else
back_id = 0;
[food,water,income] = addfw(food,water,income,restday,initwat,initfood,back_id,index);
index_day = index_day + 1;
[food,water,action,index_day] =
    walking(food,water,dist_dest,index_day,weamat,action,period,index);
dest_id = 1;
end
restday = period - index_day;
end

initwat = [206,206,206];%inital condition of water and food, respectively
initfood = [291,291,291];
mine_day = 10;
vill_day = 5;
p1 = 0.7; %possibility of sunny
p2 = 0.05; % possibility of storm
iteration = 1000;
[action,income] = sol_a(weather(p1,p2,30),initwat,initfood,30,30,mine_day,vill_day);



function [action,income] = sol_a(weamat,water,food,time,period,mine_day,vill_day)
%water and food, units of bringing, minday minimun day to go back, period
%is the total days within the dessert
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
action = zeros(3,time);
for i = 1:time
```

```
action(i) = -1;
end
income_cell = 10000;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
food_cost = food*food_price;
water_cost = water*water_price;
income_cell = income_cell - food_cost - water_cost;
income = [income_cell,income_cell,income_cell];
walk_count = 0;
initid = 1;
for i = 1:period
if weamat(i) == 0
food = food - walkrat*hitemp_unit_food;
water = water - walkrat*hitemp_unit_water;
walk_count = walk_count + 1;
action(1,i) = 1;
[action,initid,villid,villid2] = walking_decision(weamat,action,i,water,food,initid);
if villid == 3
[water,food] = costfun(water,food,weamat(i),1,2);
else
[water,food] = costfun(water,food,weamat(i),0,2);
end
if villid2 == 3
[water,food] = costfun(water,food,weamat(i),1,3);
else
[water,food] = costfun(water,food,weamat(i),0,3);
end
elseif weamat(i) == 1
food = food - walkrat*sunny_unit_food;
water = water - walkrat*sunny_unit_water;
walk_count = walk_count + 1;
action(1,i) = 1;
[action,initid,villid,villid2] = walking_decision(weamat,action,i,water,food,initid);
if villid == 3
[water,food] = costfun(water,food,weamat(i),1,2);
else
[water,food] = costfun(water,food,weamat(i),0,2);
end
```

```
if villid2 == 3
[water,food] = costfun(water,food,weamat(i),1,3);
else
[water,food] = costfun(water,food,weamat(i),0,3);
end
else
food = food - storm_unit_food;
water = water - storm_unit_water;
action(1,i) = 0;
[action,initid,villid,villid2] = walking_decision(weamat,action,i,water,food,initid);
if villid == 3
[water,food] = costfun(water,food,weamat(i),1,2);
else
[water,food] = costfun(water,food,weamat(i),0,2);
end
if villid2 == 3
[water,food] = costfun(water,food,weamat(i),1,3);
else
[water,food] = costfun(water,food,weamat(i),0,3);
end
end
if walk_count == dist_mine
break;
end
end
restday = period - i;
index_day = i;
[food,water,action,index_day,income] =
    decision_making(food,water,weamat,action,restday,index_day,income,period,water,food,mine_day,vill_day,1);
restday = period - i;
index_day = i;
[food,water,action,index_day,income] =
    decision_making_2(food,water,weamat,action,restday,index_day,income,period,water,food,mine_day,vill_day);
index_day
income = sum(income) + sum(food)*food_price/2 + sum(water)*water_price/2;
end

function [id] = mine_storage_check(food,water,current_wea,restday,mine_day,index)
%return 0 if should go to village, 2 if should directly to destination, 1
%means continue mining, restday including the day input
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
```

```
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
if current_wea == 0
watrat = hitemp_unit_water;
foodrat = hitemp_unit_food;
elseif current_wea == 1
watrat = sunny_unit_water;
foodrat = sunny_unit_food;
else
watrat = storm_unit_water;
foodrat = storm_unit_food;
end
remain_food = food(index) - craftrat*foodrat;
remain_water = water(index) - craftrat*watrat;
if restday > mine_day %value may be modified, greater than 7
if remain_food >= (2 * hitemp_unit_food * walkrat + storm_unit_food )&& remain_water >=
    (2*hitemp_unit_water*walkrat + storm_unit_water)
id = 1;
else
id = 0;
end
elseif restday > 4
if remain_food >= (3 * hitemp_unit_food * walkrat + storm_unit_food) && remain_water >=
    (3*hitemp_unit_water*walkrat + storm_unit_water)
id = 1;
else
id = 2;
end
else
id = 2;
end

end

function [food,water,action,index_day,income] =
    decision_making_2(food,water,weamat,action,restday,index_day,income,period,initwat,initfood,mine_day,vill_d
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
storm_unit_water = 10;
storm_unit_food = 10;
```

```
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
flag = -1;
record = -1;
initid = 0;
for j = index_day+1:period
for index = 2:3
[action,initid,villid1,villid2] =
    walking_decision(weamat,action,index_day,water,food,initid,mine_day);
if index == 2
decision = villid1;
else
decision = villid2;
end
if j < flag && index == record
continue;
elseif decision == 0
[food,water,action,j] = walking(food,water,dist_mine_vill,j,weamat,action,period,index);
restday = period - j;
[income,food,water,restday,action,j,dest_id] =
    vill_decision(income,food,water,restday,weamat,j,action,period,initwat,initfood,vill_day,index);
flag = j+1;
record = index;
if dest_id == 1
flag = -1;
break;
end
elseif decision == 2
[food,water,action,j] = walking(food,water,dist_dest,j,weamat,action,period,index);
restday = 0;
break;
elseif decision == 1
[water,food] = costfun(water,food,weamat(j),2,index);
action(index,j) = 2;
restday = period - j;
income(index) = income(index) + 1000;
elseif decision == 3
[water,food] = costfun(water,food,weamat(j),1,index);
action(index,j) = 1;
restday = period - j;
elseif decision == -1
```

```
[water,food] = costfun(water,food,weamat(j),1,index);
action(index,j) = 0;
restday = period - j;
end
end
index_day = j;
end


function [food,water,action,index_day,income] =
    decision_making(food,water,weamat,action,restday,index_day,income,period,initwat,initfood,mine_day,vill_day
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
flag = -1;
for j = index_day+1:period
decision = mine_storage_check(food,water,weamat(j),restday,mine_day,index);
if j < flag
continue;
elseif decision == 0
[food,water,action,j] = walking(food,water,dist_mine_vill,j,weamat,action,period,index);
restday = period - j;
[income,food,water,restday,action,j,dest_id] =
    vill_decision(income,food,water,restday,weamat,j,action,period,initwat,initfood,vill_day,index);
flag = j+1;
if dest_id == 1
flag = -1;
break;
end
elseif decision == 2
[food,water,action,j] = walking(food,water,dist_dest,j,weamat,action,period,index);
restday = 0;
break;
elseif decision == 1
[water,food] = costfun(water,food,weamat(j),2,index);
action(index,j) = 2;
```

```
restday = period - j;
income(index) = income(index) + 1000;
end
end
index_day = j;
end

function [water,food] = costfun(water,food,current_wea,status,index)
%status- 0 stand, 1 walk, 2 mining, weather - 0 high temp, 1 sunny, 2 storm
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;

if current_wea == 0
watrat = hitemp_unit_water;
foodrat = hitemp_unit_food;
elseif current_wea == 1
watrat = sunny_unit_water;
foodrat = sunny_unit_food;
else
watrat = storm_unit_water;
foodrat = storm_unit_food;
end

if status == 0
costrat = 1;
elseif status == 1
costrat = walkrat;
else
costrat = craftrat;
end

fdamt = food(index) - costrat * foodrat;
food(index) = fdamt;
wtamt = water(index) - costrat * watrat;
water(index) = wtamt;
```

```
end

function [food,water,income] = addfw(food,water,income,restday,initwat,initfood,back_id,index)
dist_mine = 5;
dist_dest = 3;
dist_mine_vill = 2;
dist_vill = 5;
storm_unit_water = 10;
storm_unit_food = 10;
hitemp_unit_water = 9;
hitemp_unit_food = 9;
sunny_unit_water = 3;
sunny_unit_food = 4;
craftrat = 3;
walkrat = 2;
food_price = 10;
water_price = 5;
food_weight = 2;
water_weight = 3;
fa = food(index);
wa = water(index);
if back_id == 1
[wa,fa] = costfun(wa,fa,0,1,index);
cost_food = (food(index) - fa)*5 + (restday - 5)*craftrat*storm_unit_food;
cost_water = (water(index) - wa)*5 + (restday - 5)*craftrat*storm_unit_water;
if (cost_food * food_weight + cost_water * water_weight) > 1200
costratio = (initfood - food(index)) / (initwat - water(index));
food_final = costratio/(1+costratio) * 1200;
water_final = 1/(1+costratio)*1200;
food_final = floor(food_final / food_weight);
water_final = floor(water_final / water_weight);
income(index) = income(index) - 2*(food_final - food(index))*food_price - 2*(water_final -
    water(index))*water_price;
food(index) = food_final;
water(index) = water_final;
else
food_add = cost_food - food(index);
water_add = cost_water - water(index);
income(index) = income(index) - 2*food_add*food_price - 2*water_add*water_price;
food(index) = cost_food;
water(index) = cost_water;
end
else
[wa,fa] = costfun(wa,fa,0,1,index);
cost_food = (food(index) - fa)*3 + storm_unit_food;
cost_water = (water(index) - wa)*3 + storm_unit_water;
```

```
food_add = cost_food - food(index);
water_add = cost_water - water(index);
income(index) = income(index) - 2*food_add*food_price - 2*water_add*water_price;
food(index) = cost_food;
water(index) = cost_water;
end
end
```

附录 E　支撑材料列表

1. 第一关.xlsx 为第一关玩家的最优策略
2. 第二关.xlsx 为第二关玩家的最优策略
3. 1.txt 为第一关中所提供的输入数据和简化图的结构
4. 2-1.txt 为第二关中第一个子图的结构以及天气等输入数据
5. 2-2.txt 为第二关中第一个子图的结构以及天气等输入数据
5. 2-3.txt 为第二关中第一个子图的结构以及天气等输入数据
6. main.cpp 为第一问(第一、二关)所用的动态规划的程序
7. "第三关"文件夹中matlab程序为第三关模拟程序，其中test.m为主函数
8. "第四关"文件夹中matlab程序为第六关模拟程序，其中test1.m为主函数
9. "第六关"文件夹中matlab程序为第六关模拟程序，其中test1.m为主函数
10. 4-模拟.xlsx，为第四关收益随晴天概率从0-1的分布