基于赋权整数优化和非合作博弈的穿越沙漠游戏攻略

摘要

“穿越沙漠”是一款与图论和运筹学知识相结合的趣味游戏，要求玩家在沙漠中依靠水和食物的补充，满足一系列约束条件，顺利走出沙漠，给出实现剩余金钱最大化的决策。本文主要基于图论、运筹优化和博弈论的相关知识针对“穿越沙漠”游戏中天气已知和未知、单人和多人的情况进行可行策略分析、求解与优化，获得了一系列使得剩余金钱最大化的决策方案。

在第一问的设定中，给定了游戏空间的天气情况，让我们给出已知天气情况下的最优调度策略。采矿的收益与长距离调度的成本相互制衡，因此第一、二关涉及多种采矿、补给策略的之间的组合优化问题，我们结合最优化理论中的整数优化方法，将采矿天数、水和食物的购买数量抽象为待优化变量、将剩余金钱抽象为收益函数，利用 Matlab 软件进行最大规划；考虑到地图不同，矿山与村庄的位置关系也不同，因此我们针对物资补充次数进行了有效讨论，建立了基于物资补充次数的单目标整数规划模型，并且给出了第一、二关在 0 到 2 次物资补充情况下的最佳决策。

在第二问中，题目设定玩家仅在当天知道当日的天气，要求我们进一步给出未知天气分布的情况下的最优决策。这是多变量之间的运筹优化问题，是第一问的延伸，本着简化模型的原则，我们预先设定两种或三种天气的先验概率分布，基于给定先验概率采用 Matlab 软件进行模拟仿真，并第四关中首次引入了赋权法求解多目标优化问题，并从决策方案中抽象出一维背包问题，求出了天气未知情况下的最佳调度决策。

沙漠的资源总是有限的，第三问在单人游戏的基础上引入了其他玩家，设定当多人在同一地点调度时消耗翻倍，同时采矿则收入减少，要求我们给出一般情况下玩家应采取的策略。我们认为，其他玩家的引入是在前两问的最佳决策上施加的非完全随机扰动，因此需要给出泛化能力更强的决策方法。因此我们结合博弈论的知识建立了基于非合作随机博弈的竞争模型，给出了双方博弈情况下的 Nash 均衡，对三方随机博弈情形下的 Nash 均衡做出了合理推断，并且基于 Nash 均衡在前四关的基础之上对前两问提出的模型进行了改进，给出了一般玩家在第五、六关获胜的调度策略。

关键词：图论　整数规划　先验概率　赋权法　非合作博弈

1 问题的背景重述

“穿越沙漠”是一款基于图论的策略规划游戏。沙漠是游戏设计者虚拟出的拓扑空间，在本游戏中，要求玩家在截止时间之前成功离开沙漠，凭借虚拟资金购买水和食物，作为穿越沙漠的物资储备；每轮游戏的剩余资金是抽象化的得分值，是评价决策的标准。游戏规则简要介绍如下：

1. 虚拟时空以“天”为基本时间单位，玩家需在规定时间前离开沙漠；

2. 玩家可在初始阶段购买水和食物，也可以在经过的村庄补充，物资具有最大负载约束；剩余的水和食物可以折半退回；

3. 三种天气情况下物资消耗具有相异性，三种天气对于游戏空间具有一致性；

4. 玩家可以前进、可停留，若停留在矿山则可在第二天挖矿。

在问题一中，要求在已知全部天气的前提下，建立数学模型，求解附件一中的第一关和第二关，完成单个玩家的最优路径规划；

相似的，在问题二，设定仅知道行程当日的天气条件，求解附件一中的第三关和第四关；

问题三是基于 n 个玩家的路径规划问题。在该前提下，k 位玩家同时从 A 地前往 B 地所需物资增长为 $2k$ 倍，挖矿收益减少为 $\frac{1}{k}$ ，同样在已知全部天气和仅知当日天气的两种情况下分别建立静态和动态的数学模型，求解附件一中的第五关和第六关，给出一般玩家的决胜策略。

2 问题分析

问题一是典型的有约束单目标整数规划模型，针对问题一采矿收益与物资消耗相互抵消的特点，由此产生的收益与损失不匹配的问题，应当将图论模型和简单整数优化模型相结合，并采用软件求解多路径组合优化下的效益最大化问题，方便玩家针对不同场景推定最优决策方案。

问题二是多变量组合优化问题，由于天气变量变为未知，因此应当给定天气的先验分布对天气进行模拟、采用贪心策略求解赋权图，并且利用赋权法求解多目标优化问题，获得泛化能力强于整数规划模型的决策模型。

对于问题三，考虑到随机天气的生成和其他玩家决策的不可预知性，以及多人游戏与生俱来的适者生存特性，应当将博弈论理论引入到游戏决策过程中，并且考虑将非合作博弈与随机博弈相结合，以生成一种具有较强鲁棒性的决策模型。

3 基本假设

1. 假设本游戏的得分与剩余资金成正相关，剩余资金越多，得分越高，对应的决策方案越好。
2. 除非特殊说明，假设沙暴天气按照极小概率生成，晴朗和高温天气按照均匀分布生成。
3. 假设 n 个玩家博弈的决策方案是理性的、完全利己的，即理性人假设。

4. 假设 n 个玩家生成的决策非完全随机决策。
5. 假设 n 个玩家生成决策的过程相互独立、互不影响。

4 符号约定

符号	含义
$G_i(V_i, E_i)$	第 i 关的无向图
$e_i(u, v)$	第 i 个无向图的边集
f	最终收益函数
f_0	启动资金
f_c	损失函数
a_j	负载成本参量
b_j	金钱成本参量
c_m	采购成本参量
d_j	初始采购箱数
d_{ij}	第 i 次采购箱数
N_n	各状态最短路径数
t_0	最大游戏天数
t_{min}	完成除采矿之外最小所需天数天数
ω_{il}	天气成本因子
p	天气状况系数矩阵
q	单一天气决策矩阵
T	目标系数矩阵
λ_i	权系数
A_i, a_j	决策集及决策空间
$D(j)$	一般玩家与其他玩家的曼哈顿距离

5 模型的建立与求解

5.1 问题一

5.1.1 模型准备

针对问题一，为了简化运算，使用 Mathematica 软件绘制无向图 $G_1(V_1, E_1)$ 和 $G_2(V_2, E_2)$ 如下：

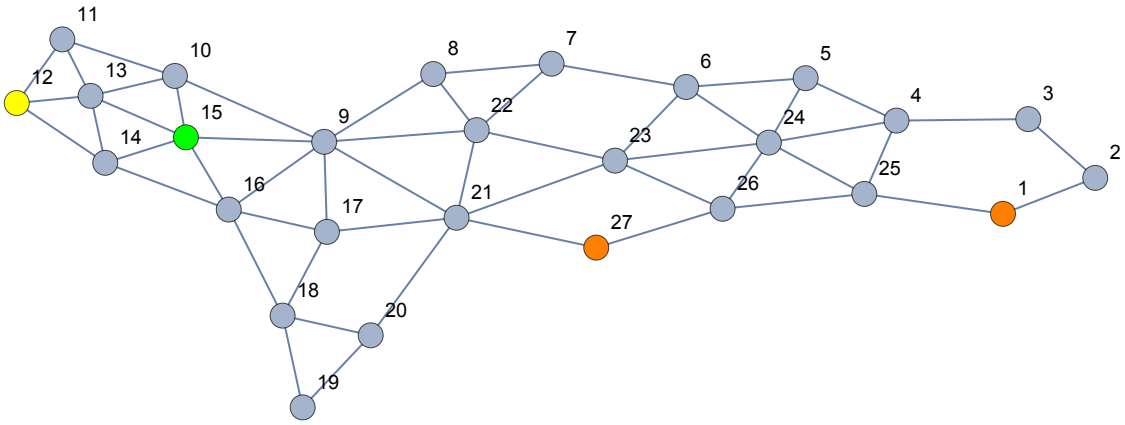

图 1: 第一关的无向图

无向图1中，顶点集 $V_1 = \{v_j | j = 1, ..., 27\}$ 中的元素在图中以数字 j 表示，起点、终点分别为 v_1 和 v_{27}，在图中以橙色顶点标识；矿山为 v_{12}，以黄色顶点标识；村庄为 v_{15}，以绿色顶点标识；其余顶点均为沙漠。

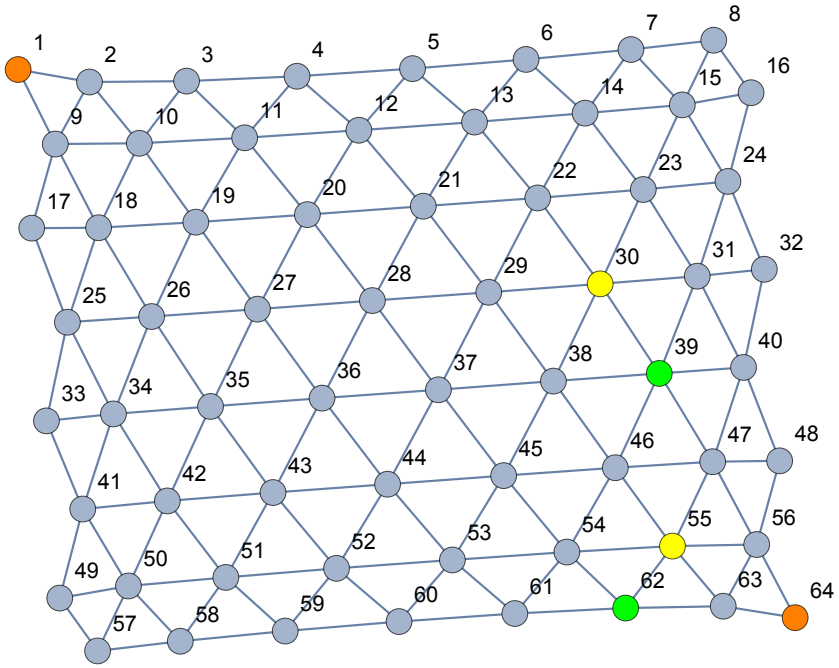

图 2: 第二关的无向图

相似地，在无向图2中，顶点集 $V_2 = \{v_k | k = 1, ..., 64\}$ 中的元素在图中以数字 k 表示，起点、终点分别为 v_1 和 v_{64}，在图中以橙色顶点标识；矿山为 v_{30} 和 v_{55}，以黄色顶点标识；村庄为 v_{39} 和 v_{62}，以绿色顶点标识；其余顶点均为沙漠。建立无向图可以直观地指导我们寻求最短路径的过程。

5.1.2 基于物资补充次数的单目标整数规划模型

由图1及2可知，玩家从起点到终点有多条“最短路径”可以通过。考虑从起点到终点的最短距离，可以采取“起点 → 终点”的最短距离策略，此策略可以实现物料消耗最小化，从而达成最大化剩余资金的目标；考虑从矿山采矿获得资金，可以采取“起点 → 采矿 → 终点”的采矿收益最大化策略；在实际情况下，考虑到玩家前往矿山及采矿本身对水和食物等物资的消耗，可以采取“起点 → 采矿 → 资源补充 → 终点”的收益最大化策略，也可以采取“起点 → 资源补充 → 采矿 → 终点”的策略。考虑到长距离运输下，玩家消耗的物料可能与采矿收益相抵消，使得收益效率降低、甚至产生负效益的情况，我们以最大化剩余资金为优化目标，建立基于采矿天数和物料补充次数的单目标整数规划模型。

具体各阶段的优化流程如下：

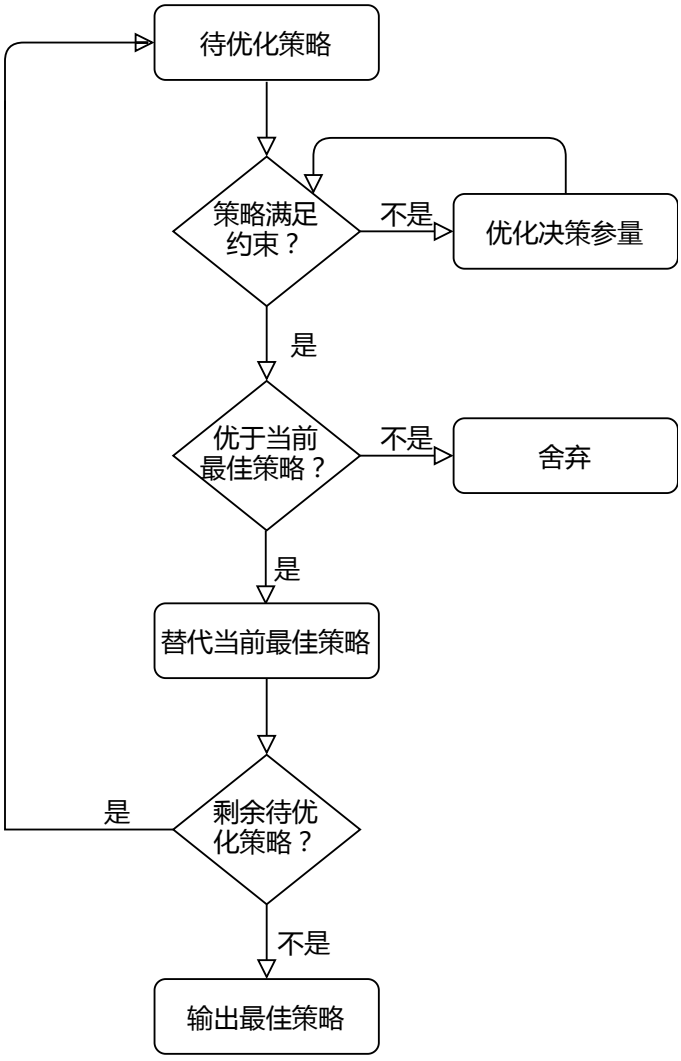

图 3: 优化流程

首先考虑玩家不进行物资补充的情形。在这种情况下，玩家可以不进行采矿，直接由起点到终点，也可以在满足物资约束的情况下采矿，以期增加最终的收益。我们不妨用最终收益函数 f 表示游戏结束时的剩余资金，量化游戏得分。

不妨假定 $a_j, b_j, j = 1,2$ 分别为负载成本参量和金钱成本参量，j 的两个取值分别指代水和食物的负载成本与金钱成本；$c_m, m = 1,2$ 是采购成本参量，当在村庄采购时 $c_2 = 2$；ω_{il} 为天气成本因子，$i = 1,..,n$ 指代游戏天数，$l = 1,2,3$ 分别指代晴朗、高温、沙暴三种天气；$d_j, j = 1,2$ 表示水和食物的采购箱数；a_0 为最大负载。

同时，假定 α_k 为状态因子，其取值为

$$\alpha_k = \begin{cases} 1, k = 1 \\ 2, k = 2 \\ 3, k = 3 \end{cases}$$

当 $k = 1$ 时为静止过程，$k = 2$ 时为行进过程，$k = 3$ 时为采矿过程。

在不采矿的策略下，我们可以直接用从起点到终点的最短路径计算收益函数 f 得：

$$f = f_0 - \sum_{j=1}^{2} c_1 b_j d_j$$

在采矿的策略下，假定 f_i 为每日采矿收益，对采矿天数 x 进行单目标决策优化：

$$max \quad f = f_0 + f_i x - f_c$$

$$s.t. \quad \begin{cases} d_{0j} \leqslant \sum\limits_{j=1}^{2} b_j c_1 d_j \leqslant f_0 \\ 0 \leqslant \sum\limits_{j=1}^{2} a_j c_1 d_j \leqslant a_0 \\ 0 \leqslant x \leqslant t_0 - t_{min} \\ x \in \mathbb{N} \end{cases}$$

其中，d_{0j} 表示从起点到终点的最小物资消耗；损失函数 $f_c = \sum\limits_{i=1}^{n} \alpha_k \omega_{il} b_j c_m, i = 1,...,n; j = 1,2; k = 1,2,3; l = 1,2,3; m = 1,2$ 表示静止、前进和挖矿的消耗物资产生的金钱损失。

当考虑一次补给时，实际存在的决策方案可能有：由起点直接到达终点；起点 $\rightarrow$ 补给 $\rightarrow$ 采矿 $\rightarrow$ 终点；起点 $\rightarrow$ 采矿 $\rightarrow$ 补给 $\rightarrow$ 终点等。考虑实际情况，针对常见的游戏场景，不妨只考虑后两种情况下的目标优化模型。

当考虑起点 $\rightarrow$ 补给 $\rightarrow$ 采矿 $\rightarrow$ 终点策略时，同样假定 f_i 为每日采矿收益，对采矿

天数 x、初始额外购买物资箱数 y_1, y_2、物资补充数 z_1, z_2 进行单目标决策优化：

$$
\begin{aligned}
max \quad & f = f_0 + f_i x - f_c \\
s.t. \quad & \begin{cases}
d_{0j} \leqslant \sum_{j=1}^{2} a_j c_1 d_j \leqslant a_0 \\
0 \leqslant \sum_{j=1}^{2} b_j c_2 z_j \leqslant f_0 - \sum_{j=1}^{2} b_j c_1 d_j \\
d_{1j} \leqslant \sum_{j=1}^{2} a_j c_2 z_j \leqslant a_0 - \sum_{j=1}^{2} a_j c_1 (d_j - d_{0j}) \\
0 \leqslant x \leqslant t_0 - t_{min} \\
x, y_1, y_2, z_1, z_2 \in \mathbb{N}
\end{cases}
\end{aligned}
$$

其中 $d_j = d_{0j} + y_j, j = 1, 2$，$d_{0j}$ 为从起点到补给的村庄的水和食物的最小消耗量；假定从村庄完成采矿和返回终点的任务天数为 N，d_j 为从村庄完成采矿过程的水和食物的最小消耗量；f_c 为损失函数，计算方法同上。

当考虑起点 $\rightarrow$ 采矿 $\rightarrow$ 补给 $\rightarrow$ 终点策略时，同样针对以上 5 个参数进行单目标决策优化：

$$
\begin{aligned}
max \quad & f = f_0 + f_i x - f_c \\
s.t. \quad & \begin{cases}
0 \leqslant \sum_{j=1}^{2} b_j c_1 d_j \leqslant f_0 \\
d_{0j} \leqslant \sum_{j=1}^{2} a_j c_1 d_j \leqslant a_0 \\
0 \leqslant \sum_{j=1}^{2} b_j c_2 z_j \leqslant f_0 + f_i x - \sum_{j=1}^{2} b_j c_1 d_j \\
d_{1j} \leqslant \sum_{j=1}^{2} a_j c_2 z_j \leqslant a_0 - \sum_{j=1}^{2} a_j c_1 (d_j - d_{0j}) \\
0 \leqslant x \leqslant t_0 - t_{min} \\
x, y_1, y_2, z_1, z_2 \in \mathbb{N}
\end{cases}
\end{aligned}
$$

其中 $d_j = d_{0j} + y_j, j = 1, 2$，$d_{0j}$ 为从起点采矿并前往村庄补给的最小消耗量；假定从村庄返回终点的天数为 N,d_j 为从村庄返回终点的最小物资消耗量；f_c 为损失函数，计算方法同上。

在实际操作过程中，可能出现可用天数未充分利用的情形，即过早地返回终点，造成时间浪费。因此，适当增加补给次数，延长挖矿时间，可能对增加收益产生积极作用。考虑到可能增加的补给次数可能包括两次、三次、四次等，而在补给次数为 2 时，就已经产生了至少 $C_4^2 + 1$ 种组合情形。基于游戏现实，接下来我们仅针对二次补给下一个常用的情况进行建模。

考虑补给 $\rightarrow$ 采矿 $\rightarrow$ 补给 $\rightarrow$ 采矿情形，对采矿天数 x_1, x_2、初始额外购买物资箱

数 y_1, y_2、第一次物资补充数 z_1, z_2、第二次物资补充数 u_1, u_2 进行单目标决策优化：

$$
\begin{aligned}
max \quad & f = f_0 + f_i x - f_c \\
s.t. \quad & \begin{cases}
0 \leqslant \sum_{j=1}^{2} b_j c_1 d_j \leqslant f_0 \\
d_{0j} \leqslant \sum_{j=1}^{2} a_j c_1 d_j \leqslant a_0 \\
0 \leqslant \sum_{j=1}^{2} b_j c_2 z_j \leqslant f_0 - \sum_{j=1}^{2} b_j c_1 d_j \\
d_{1j} \leqslant \sum_{j=1}^{2} a_j c_2 z_j \leqslant a_0 - \sum_{j=1}^{2} a_j c_1 (d_j - d_{0j}) \\
0 \leqslant \sum_{j=1}^{2} b_j c_2 u_j \leqslant f_0 + f_i x_1 - \sum_{j=1}^{2} b_j c_1 d_j - \sum_{j=1}^{2} b_j c_2 z_j \\
d_{2j} \leqslant \sum_{j=1}^{2} a_j c_2 u_j \leqslant a_0 - \sum_{j=1}^{2} a_j c_2 z_j \\
0 \leqslant x \leqslant t_0 - t_{min} \\
x, y_1, y_2, z_1, z_2, u_1, u_2 \in \mathbb{N}
\end{cases}
\end{aligned}
$$

其中其中 $d_j = d_{0j} + y_j, j = 1, 2$，$d_{0j}$ 为从起点完成第一次采矿动作并前往村庄补给的最小消耗量；d_{1j} 为从村庄返回矿山采矿并再次返回村庄的最小物资消耗量；d_{2j} 为完成第二次采矿后从矿山返回终点的最小消耗量；f_c 为损失函数，计算方法同上。

5.1.3 模型的求解与验证

基于上述模型，初始时刻需要满足约束条件：

$$
\begin{cases}
\sum_{j=1}^{2} a_j d_{0j} \leqslant 3x + 2y \leqslant 1200 \\
0 \leqslant 5x + 10y \leqslant 10000
\end{cases}
$$

由于在村庄获取食物和水是起始点的两倍，假设此时剩余资金为 f'，满足

$$f' = f_0 - 10d_1 - 20d_2$$

考虑到玩家游戏过程中，食物的金钱花费明显高于水的金钱花费，而村庄中食物的价格相对较贵，为了最小化开支，在优化时应尽量增加初始时刻购买食物的数量，减少去村庄购买食物的数量。

第一关：在第一关中，天气情况全部已知，只有一个矿山和一个村庄。考虑第一种情形，假设玩家采取不采矿，直接从起点前往终点的策略，利用 Mathematica 软件求得最短路径为 $v_1 \to v_{25} \to v_{26} \to v_{27}$。在这种情况下，收益函数 $f = 10000 - 2*95 - 100 = 9710$。

再考虑起点 $\to$ 补给 $\to$ 采矿 $\to$ 终点的决策方案。利用 Mathematica 软件计算从起

点 v_1 途径村庄 v_{15} 前往矿山的最短路径集为

$$v_1 \to v_{25} \to v_{24} \to v_{23} \to v_{21} \to v_9 \to v_{15} \to v_{13} \to v_{12}$$

需要在 9 个顶点中经历 8 条边。由于前 8 天中有 2 天是沙暴天气，沙暴天气必须停留，假设玩家仅在沙暴天气下停留，则实际从起点到矿山的时间为 10 天。再利用 Mathematica 软件计算从矿山 v_{12} 前往终点 v_{12} 的最短路径为

$$v_{12} \to v_{11} \to v_{10} \to v_9 \to v_{21} \to v_{27}$$

于是从矿山到终点最少经过 6 个节点，走 5 段路程，此时天气情况不确定。考虑到在任何天气下采矿都能产生正收益，因此为了能尽可能地最大化挖矿天数 x，暂且选择最后没有沙暴的五天走完矿山到终点的剩余路程，此时 30 天中还剩下 15 天。由 Mathematica 生成的最短路径如图所示：

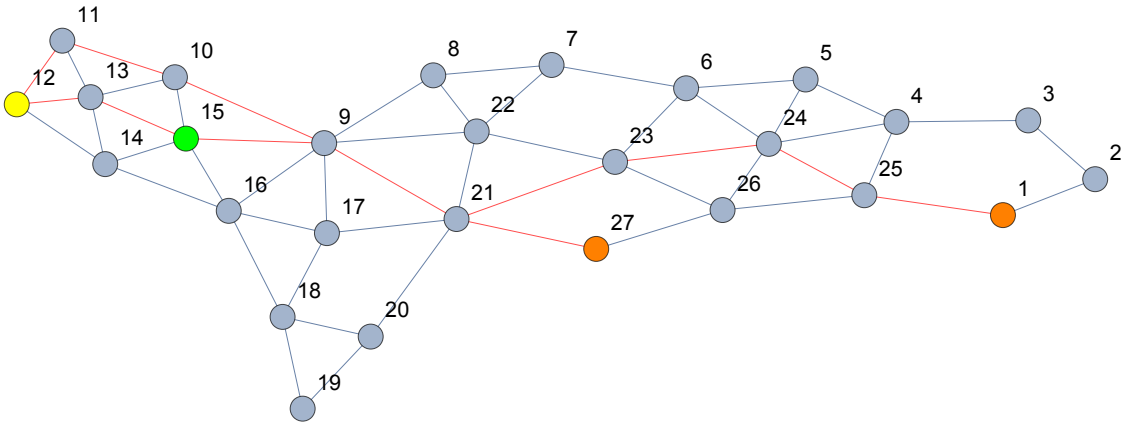

图 4: 第一关的最优决策示意图

利用 EXCEL 软件算得从起点到矿山所需补给的水为 130 箱，食物 122 箱，所需花费为 1870 元；从矿山到终点所需补给为水 78 箱，食物 74 箱，所需花费为 1130 元，这两段路程补给的负重之和为 1016 千克。在矿山挖矿的 15 天中所需食物数量为 303 箱，水数量为 335 箱。选择从起点到矿山的途中经过村庄并购买补给，从起点到村庄所需水的数量为 99 箱，所以在起点购买水 $d_1 = 99$ 箱，食物 $d_2 = 453$ 箱。所需资金为 5025 元，剩余资金 4975 元。假如剩下的 15 天全部用来挖矿，那么第一次经过村庄时购买余下剩余路程所需的食物和水所需的资金 5370 元，还差 395 元，并且决定额外在 2 个沙暴天气下休息，此时恰好满足约束条件。

最终利用 EXCEL 软件算得损失函数 $f_c = 11525$，到达终点时收益函数 $f = 13205$。

最后考虑第三种决策，即考虑两次补给的决策方案，这种情况是考虑前往矿山采矿 $\to$ 补给 $\to$ 采矿 $\to$ 补给的情形，利用 EXCEL 软件算得收益函数 $f = 11300$。

显然，第二种决策优于第一种和第三种，故采取第二种策略。

第二关：对于第二关，图中含有两座矿山两座村庄，假设决策集 $A = \{a_i\}, i = 1, 2, 3$，决策空间 a_i 分别对应不补给、一次补给、两次补给 3 种情形。由于矿山与村庄相邻，为

了简化求解过程，不妨假设玩家采矿前一定前往村庄购买物资，则情况一对应模型中无补给的情形，情况二、三对应一次补给的情形，情况四对应模型中两次补给的情形。通过模型中的整数规划解出每一种情况的最优解，比较得第二关的最佳策略。

本关中，方案 a_1 不需要补给，只有一种从起点直接到终点的情形，由于估算其收益函数 $f \leqslant 10000$，故不需计算即可舍弃。

方案 a_2 有两种情形，其中包括路过矿山 v_{30} 和矿山 v_{55} 两种情形。其中，经过 EXCEL 优化可得路过矿山 v_{55} 时其收益函数 $f = 11285$，路过矿山 v_{30} 时仅补给一次，其收益函数明显不如下面的补给两次的情形高，故舍弃。

方案 a_3 需要补给两次，情况更加复杂。首先考虑起点到矿山 v_{30} 和矿山 v_{55} 的情况，利用 Matlab 计算收益函数为 $f = 11955$。然而结合图论的知识，经过 Matlab 优化[1]，我们最终确定该情形下最优的决策方案为从起点前往矿山 v_{30} 采矿，在村庄 v_{39} 采购，返回 v_{30} 采矿，返回 v_{39} 补充并返回终点。在该方案下，利用 Matlab 计算得需要在起点购买水 $d_1 = 247$ 箱，食物 $d_2 = 229$ 箱，采矿天数 $x = 16$ 天，解得此时的收益函数 $f = 14115$。

显然，前往矿山 v_{30} 并就近补充两次的方案是当下的最优决策。利用 Mathematica 软件求得最优决策路径为

$$v_1 \to v_2 \to v_3 \to v_4 \to v_5 \to v_{13} \to v_{22} \to_{30} \to_{39} \to v_{30} \to v_{39} \to v_{47} \to v_{56} \to v_{64}$$

绘制最优决策方案下最短路径如5所示：

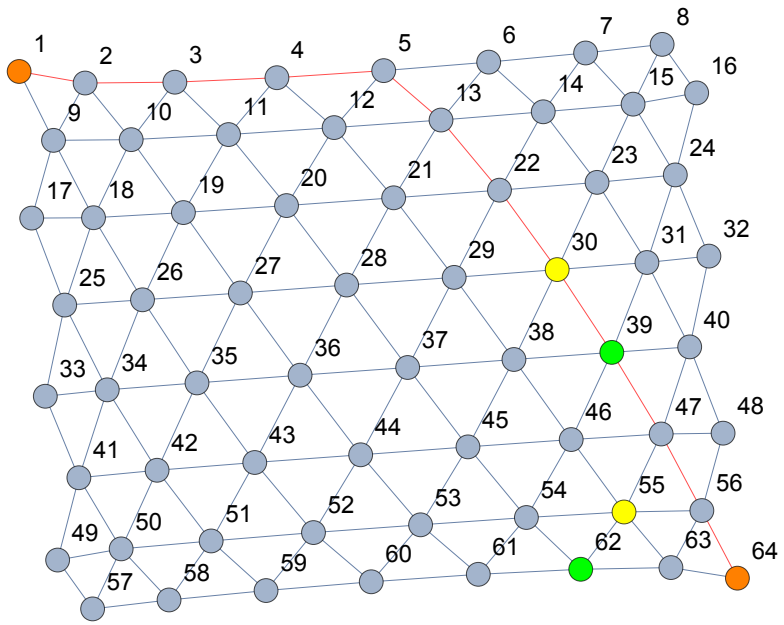

图 5: 第二关的最优决策示意图

最终优化结果如图6所示：

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	4975	99	453
1	25	4975	83	441
2	24	4975	67	429
3	23	4975	57	415
4	21	4975	47	405
5	21	4975	37	391
6	9	4975	21	379
7	15	4975	11	369
8	13	5	406	401
9	12	5	390	387
10	12	1005	366	369
11	12	1005	356	359
12	12	2005	332	341
13	12	3005	317	320
14	12	4005	293	302
15	12	5005	269	284
16	12	6005	245	266
17	12	6005	235	256
18	12	7005	205	226
19	12	8005	181	208
20	12	9005	157	190
21	12	10005	142	169
22	12	11005	127	148
23	12	12005	103	130
24	12	13005	88	109
25	12	13005	78	99
26	11	13005	62	87
27	10	13005	52	73
28	9	13005	42	59
29	21	13005	26	47
30	27	13205	10	35

第二关

日期	所在区域	剩余资金数	剩余水量	剩余食物量
0	1	6475	247	229
1	2	6475	231	217
2	3	6475	215	205
3	4	6475	205	191
4	4	6475	195	181
5	5	6475	185	167
6	13	6475	169	155
7	13	6475	159	145
8	22	6475	149	131
9	30	6475	133	119
10	30	7475	109	101
11	30	8475	79	71
12	30	9475	55	53
13	30	10475	40	32
14	30	11475	16	14
15	30	5295	210	206
16	30	5295	194	194
17	30	6295	164	164
18	30	7295	134	134
19	30	8295	118	122
20	30	9295	102	110
21	30	10295	92	96
22	30	11295	76	84
23	30	12295	66	70
24	30	13295	56	56
25	30	14295	26	26
26	30	15295	10	14
27	39	14115	42	38
28	47	14115	32	24
29	56	14115	16	12
30	64	14115	0	0

图 6: 最终优化结果

5.2 问题二

5.2.1 模型准备

问题二就问题一而言，最大的变化便是天气变量由已知转变为未知的参数，增大了问题的随机性和难度。同时，经过初步观测，我们发现第三关的图中并没有出现村庄，因此第三关不再需要考虑中途补给。因此，针对随机的天气变量，我们初步决定给定天气的分布概率建立概率模型，但路线仍考虑点到点的最短路径，并考察给定的概率分布下我们的最优决策；在第四关时，针对更复杂的地图我们将模型转化为 0-1 背包问题，并采用加权法求解多目标线性优化问题[2]。

对于问题二，我们仍然使用 Mathematica 软件对地图进行图论抽象，所得的无向图 $G_3(V_3, E_3)$ 和 $G_4(V_4, E_4)$ 如图7和8所示。

图7中，顶点集 $V_3 = \{v_j | j = 1, ..., 13\}$ 中的元素在图中以数字 j 表示，起点、终点分别为 v_1 和 v_{13}，在图中以橙色顶点标识；矿山为 v_9，以黄色顶点标识；其余顶点均为沙漠。

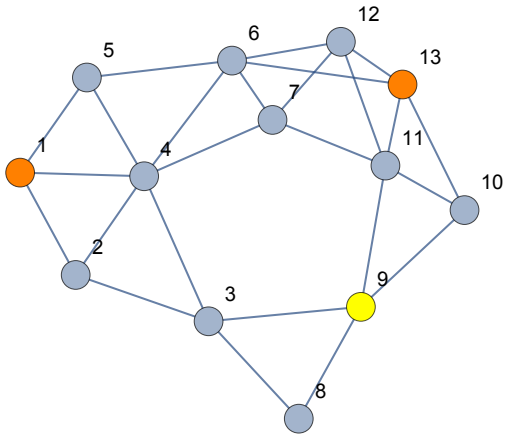

图 7: 第三关的无向图

图8中，顶点集 $V_4 = \{v_k | k = 1, ..., 25\}$ 中的元素在图中以数字 k 表示，起点、终点分别为 v_1 和 v_{25}，在图中以橙色顶点标识；矿山为 v_{19}，以黄色顶点标识；村庄为 v_{14}，以绿色顶点标识；图中其余顶点均为沙漠。

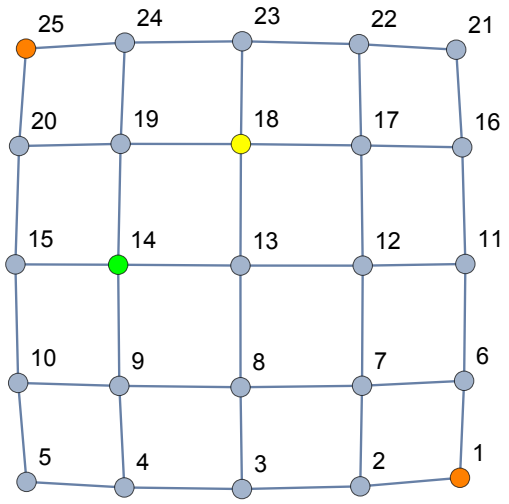

图 8: 第四关的无向图

5.2.2 基于随机天气求解最优调度策略

在第三四关中，天气情况变得未知（已知十天内不会出现沙暴天气），但是在第三关中，截止日期只有 10 天，10000 元的初始资金和 1200 千克的最大负重绰绰有余，这时仅需考虑最大化收益函数。但是这时无法从全局考虑最优解，因此采用贪心的思想：从起点开始，每一步都要确保局部最优解，在满足约束条件的情况下，当下一步的走法和前面的走法连在一起不在构成一个可行解时，就不把这一步的走法列入到最终路线中。直到达到终点为止。同时可以看出，问题二的求解仍然可以采用第一关和第二关的基本思路，即采用确保截止时间能达到终点的情况下，在矿山停留足够长的时间的规划类模型。

第三关：从第三关的无向图7中可以看出，起点 $\rightarrow$ 矿山 $\rightarrow$ 终点的最短路径为

$$v_1 \rightarrow v_4 \rightarrow v_3 \rightarrow v_9 \rightarrow v_{11} \rightarrow v_{13}$$

首先考虑前往矿山采矿的方案。为了最大化收益函数，考虑令玩家采取在前三天从起点到达矿山，在矿山停留五天，最后的两天从矿山返回终点的策略，刚好在截止时间内到达终点。在本关中，天气是未知变量，玩家只能根据当天天气情况来决定行动方案。考虑到多变量问题的求解往往需要固定某些变量，我们不妨事先给定天气的先验分布，在给定的概率分布下对天气变量进行模拟，从而降低求解难度。

假设晴朗和高温天气服从 0-1 分布，利用软件将两种天气按照各 50% 的概率生成。两种天气的分布概率图如9所示：

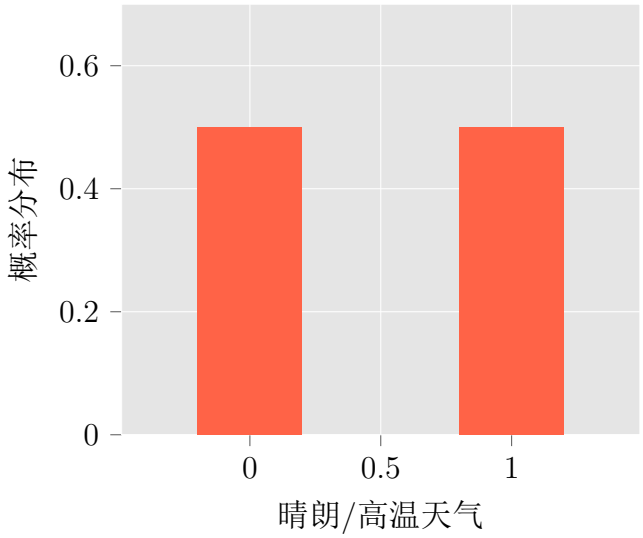

图 9: 晴朗和高温天气服从 0-1 分布

针对不同天气给出不同的策略：在路线必须经过矿山的情况下，为了保证采矿时间 x 最大，前期和后期的路程中，无论遇到晴朗抑或高温都必须行走。在矿山停留的五天中，第一天不能挖矿，而且如果在高温天气天气下采矿，采矿的金钱损耗大于不采矿的金钱损耗，所以在高温天气挖矿不合理，应当在原地休息。

与此同时，我们还可以观察到如果直接从起点到终点，不经过矿山有一条更短的路径，最少只需要 3 天即可到达。考虑到即使在金钱消耗最少的晴朗天气下，挖矿收益也十分微薄（相当每天收益 5 元），很可能不足以抵消多走的路程来带的消耗，因此可以考虑第二种方案，即直接从起点走到终点，不考虑天气带来的变化，无论晴天还是高温都行走的策略。这一策略的最短路径为

$$v_1 \rightarrow v_5 \rightarrow v_6 \rightarrow v_{13}$$

当经过矿山时，利用 Matlab 给出的一种随机模拟天气状况为 {高温, 高温, 晴朗, 高温, 高温, 晴朗, 晴朗, 高温, 高温, 高温}，按照模型一的约束，解得收益函数 $f = 8475$。

当直接由起点前往终点时，利用 Matlab 给出的一种随机模拟天气状况为 {高温, 晴天, 高温}，按照模型一的约束，解得收益函数 $f = 9350$。

同时，考虑两种极端情况（即全为晴朗或全为高温两种情况），可以计算得经过矿山的受益函数区间为 $7975, 9625$，不经过矿山的收益函数区间为 $9190, 9720$，两种方案的极差 R 分别为 1650 和 530。

这时可以看出，经过矿山的剩余资金受到天气因素影响较大，区间跨度也更大，而不经过矿山的走法则就非常保险了，并且不经过矿山的剩余资金上限比经过矿山的剩余资金上限要高。由于此时天气随机变量满足等概率 0-1 分布，两种决策下的收益函数分布应当符合正态分布，其期望 $\mu_1 = 8800 < \mu_2 = 9455$，也可推断标准差 $\sigma_1 > \sigma_2$。综上所述，不采矿、直接从起点前往终点的方案是更加合理和稳妥的，因此最优决策就是不采矿方案。

利用 Mathematica 软件绘制最佳决策路径如图10所示：

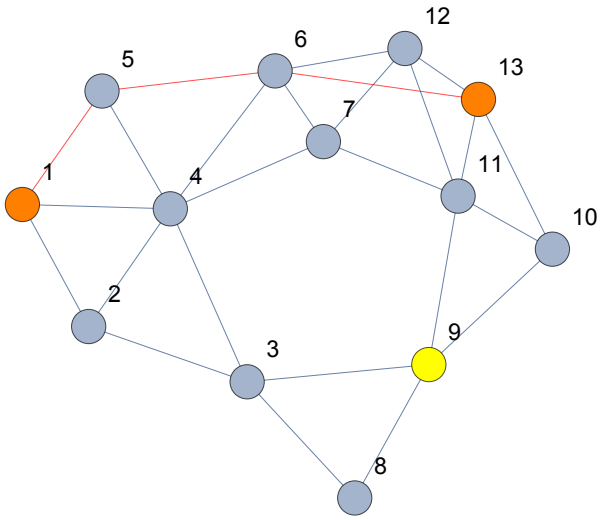

图 10: 第三关的最优决策示意图

第四关：第四关中，天气情况仍然是未知的，此时还加入了沙暴天气条件。截止日期变成 30 天，这时需要考虑食物和水的负重，但是这时在矿山挖矿得到的基础收益远大于最恶劣天气沙暴的三倍基础消耗量，所以此时直接从起点到终点显然是不合理的，我们继续考虑起点 $\rightarrow$ 矿山 $\rightarrow$ 终点的路线。

对于每一天来说，明天的天气状况是未知的，昨天的天气状况是已知的，每一天的天气状况是互不干扰的，但是为了剩余的资金足够多，同样也要保证每一步都是最优解。

题目中提到 30 天中较少出现沙暴天气，所以我们利用计算机生成 1 100 范围内的随机数，如果随机数在 $[1, 40]$ 区间内，则模拟为晴天天气，$[41, 80]$ 区间内为高温天气，否则则为沙暴天气。则三种天气的分布期望如图11所示。

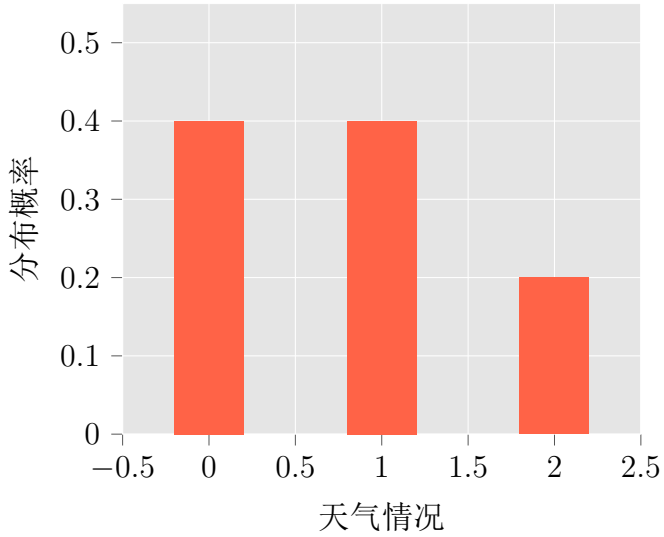

图 11: 三种天气的分布期望

此时我们可以通过加权系数法进行目标函数转换。加权系数法是一种将多目标函数转换成单目标函数的方法，通过权系数来决定目标对于决策者的重视程度，通过加权求和将多个目标函数转换成单一目标函数[3]。考虑加权系数法，我们获得目标函数：

$$Z = \lambda_1 Z_1 + \lambda_2 Z_2$$

其中 Z_1, Z_2 为完成前 n 步转移的最小水和食物消耗箱数，λ_1, λ_2 是权重系数，满足

$$\lambda_1, \lambda_2 \geqslant 0$$

$$\lambda_1 + \lambda_2 = 1.$$

关键指标在综合体系中越重要，所对应的权系数就应该越大，对于最后不影响决策的指标占全系数就应该很小。确定权系数的方法有很多，此处采用相对比较法，虽然避免不了主观随意性，但是由于目标函数之间关联较强，并不会影响最后的结果。对于这场游戏来说，食物和水对结果的影响是基本相同的，所以取权系数为 $\frac{1}{2}$。可以把目标函数中的系数看做背包问题中的价值，把约束条件中的系数看做背包问题的重量，进而用贪心算法解一维背包问题[4]。求解目标函数最大值的步骤如下：

1. 输入根据天气情况的状态系数矩阵 p，单一天气决策矩阵为 q，目标系数矩阵为 T；

2. 计算单个变量的性价比矩阵 $c = T/p$，并且把其中的元素按照从小到大的顺序得到临时矩阵 r；

3. 按照矩阵 r 中的顺序对 T, p, q 中的元素进行排序；

4. 依照贪心选择策略，尽可能选择性价比高的策略，每一次选择都要保证负重总量不超过 1200 千克，同时每一次选择都要保证食物和水不为 0，直到选择最后一个为止；

5. 依次输出目标函数的最大值，剩余的资金数，路线。

最终利用 Matlab 软件算得收益函数 $f = 16100$。

5.3 问题三

5.3.1 基于非合作随机博弈的竞争模型

在考虑多玩家因素以后，整个游戏空间可以抽象为一类非合作博弈模型，而如果假设各位玩家的游戏过程相互独立、互不干扰，整个游戏又产生了一定的随机性特征，因此不妨将其他玩家的路线视为随机扰动。考虑到每一位玩家都试图追求决策利益最大化，因此这一扰动并不是完全随机的。在非合作博弈中，如果我们要求博弈后能相较其他玩家获得更高的分数，就要求在发生非完全随机扰动的情况下追求收益函数最大化。非合作博弈 Nash 均衡理论对于解决生存与策略调度优化问题有独到的效用，结合前面两问的成果，我们进一步提出了基于非合作随机博弈的竞争模型。

第五关：考虑仅有一座矿山、没有村庄，有且仅有两个玩家的情形。假设两位玩家的决策过程完全独立，但不允许采取完全随机决策，那么不妨假设两位玩家的决策集 $A_1 = A_2 = \{a_1, a_2\}$，其中决策空间 a_1 为玩家考虑起点 $\rightarrow$ 矿山 $\rightarrow$ 终点的最短路径，并考虑最大化采矿时间、最小化初始阶段物资消费的决策，空间 a_2 为玩家考虑不采矿，直接从起点到终点的的决策，绘制对于一般玩家而言的博弈支付矩阵如表1：

表 1: 决策集 A_1, A_2 的博弈支付矩阵

		a_2	
		采取	不采取
a_1	采取	$cost_1$	$cost_2$
	不采取	$cost_2$	$cost_3$

如果此时假设策略 a_1 的收益远大于 a_2，则由问题一和二以及经验推理，可得 $cost_2 > cost_3 > cost_1$，则两位玩家非合作博弈的 Nash 均衡，就是两位玩家均采取 a_1 策略。如果两位玩家同时严格执行这一策略，且在一方留在原地时另一方也随之停留，则博弈的结果可能是平局；如果一方采取这一策略，而另一方不考虑这一策略，则不考虑这一策略的一方会由于采矿收益较另一方少，或者是路途中消耗水和食物较另一方多而失败。当然，这是基于采矿收益远大于起点 $\rightarrow$ 终点最短路径策略的假设产生的 Nash 均衡，假如相比采矿策略，采取不采矿策略能获得更高的分数，那么该情况下两位玩家非合作博弈的 Nash 均衡就是均采取直接从起点到终点的不采矿策略。

第六关：考虑第六关中仅有一座矿山和村庄，且有三个玩家的情形。由于模型考虑三方博弈，且存在随机博弈的特征，模型的 Nash 均衡很难确定是否存在[5]。但是不妨由第五关推理，假设这一模型的 Nash 均衡点也存在，那么要么是多位玩家均采取沿最短路径采矿并返回的策略，要么是均采取起点 $\rightarrow$ 终点策略。假设此时收益函数 $f_1 >> f_2$，为了简化模型，我们不妨假设所有玩家仅仅在问题一中提及的无补给、一次补给、二次

补给中选择策略。在这种情形下，我们不妨建立以起点 v_1 为坐标原点，以水平向右和竖直向上为正方向建立二维笛卡尔坐标系，并借鉴曼哈顿距离的概念，定义各个节点的曼哈顿距离为 2，来考虑一般玩家与第 j 位玩家的距离：

$$D_j = |x - x_j| + |y - y_j|, j = 1, 2.$$

绘制坐标系如图12所示：

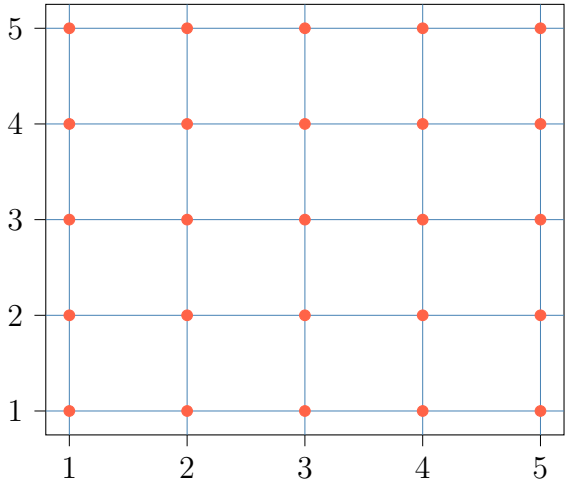

图 12: 抽象的二维笛卡尔坐标系

比如，在无向图 $G_4(V_4, E_4)$ 中，起点与村庄的最短路径长为 6，因此两者之间的曼哈顿距离也就为 6。考虑到在多位玩家同时购买食物时金钱成本参量 $bj = 4$，那么一般玩家应该在第 7 天或第 8 天到达村庄，避免扎堆。为了能实现路径最短，应当限定玩家与目标点的距离

$$D_{(x,y)\to target} \leqslant D_k, k = 1, 2, ...$$

其中 D_k 为出发点与目标点之间的最短距离，且要求每次移动都要求距离缩小，并允许松弛一次或两次，作为留在原地不动或不得已朝向目标反方向行走的缓冲，这样就满足了在第 7、8 天到达村庄的约束。另外，不妨借鉴元胞自动机中 Von Neuman 型邻域的概念规划路径[6]。元胞自动机中的 Von Neuman 型邻域如13所示。

由图可知，Von Neuman 型邻域仅仅包含了距离中心点保持曼哈顿距离为 1 的 4 个点。当前状态下，当一般玩家正处于其他玩家的 Von Neuman 型邻域中，如果原地不动，则下一时刻有 $\frac{1}{4}$ 的概率和其他玩家同时落脚同一区域，考虑 Nash 均衡，玩家应当向目标方向移动一次；当玩家的下步转移点与与其他玩家的 Von Neuman 型邻域重合，如果仍向该点转移，则会有 $\frac{1}{4}$ 的概率与其他玩家重合，按照 Nash 均衡原则，应当将该节点封禁；而如果当前玩家的落脚点已经与其他玩家重合，考虑 Nash 均衡，也应当向目标方向移动一次。

另外，仍然应当尽量最大化矿山采矿的天数。对于第六关中天气未知的难题，可以采用给定天气的先验概率分布并采用蒙特卡罗方法模拟的方法，这样就形成了能够自演化的基于非合作随机博弈的竞争模型。

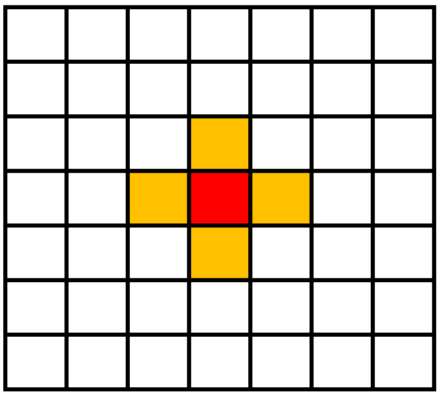

图 13: Von Neuman 型邻域结构

6 模型评价与改进

6.1 模型的优点

1. 核心采用单目标整数规划方法，模型经典，通俗易懂，增强了模型的鲁棒性，泛化能力较强。
2. 通过给定先验概率的方法仿真模拟真实游戏场景，降低了模型的难度，且便于向其他研究方向推广。
3. 以非合作博弈理论解决运筹优化问题，使抽象的问题形式一般化，且增强了求解过程的创新性。

6.2 模型的改进

1. 考虑到决策方案空间过于复杂，对模型做出了不少假设，可以在仿真时适当减少假设，以考虑更多排列组合，并行采用遗传、模拟退火等智能优化算法寻优。
2. 真实游戏中的天气变化更加随机化，可以考虑对天气按照不同随机分布进行讨论，并且采用蒙特卡罗方法进行仿真，增加试验次数，减少系统误差。
3. 对于第三关，考虑图论中流量与成本优化问题，还可以采用经典的最小费用最大流算法。

7 参考文献

[1] 王海英，黄强，李传涛，褚宝增. 图论算法及其 MATLAB 实现 [M]. 北京：北京航空航天大学出版社，2010.

[2] 司守奎，孙玺菁等. 数学建模算法与应用 [M]. 北京：国防工业出版社，2011.

[3] 周兵, 王传生, 刘芳亮. 基于大型桥梁的最优交通流量控制策略选择 [J]. 长安大学学报 (自然科学版),2020,40(04):68-77.

[4] 袁绛书, 冯振宇, 朱天乐, 陈铭思, 姜雨辰. 基于贪心算法的黄山景区旅游路线优化设计 [J]. 现代商贸工业,2020,41(20):32-33.

[5] 随机博弈 | 机器之心 [OL].https://www.jiqizhixin.com/graph/technologies/014952e9-

1292-4791-9b5a-348a4547ddb6
[6] 段晓东, 王存睿, 刘向东. 元胞自动机理论研究及其仿真应用 [M]. 北京: 科学出版社,2012.

8 附录

8.1 附录 A——求解两点最短距离的 Matlab 源程序

```
function [P u]=n2shorf(W,k1,k2)
n=length(W);
U=W;
m=1;
while m≤n
    for i=1:n
        for j=1:n
            if U(i,j)>U(i,m)+U(m,j)
            U(i,j)=U(i,m)+U(m,j);
            end
        end
    end
m=m+1;
end
u=U(k1,k2);
P1=zeros(1,n);
k=1;
P1(k)=k2;
V=ones(1,n)*inf;
kk=k2;
while kk≠k1
    for i=1:n
        V(1,i)=U(k1,kk)-W(i,kk);
        if V(1,i)==U(k1,i)
            P1(k+1)=i;
            kk=i;
            k=k+1;
        end
    end
end
k=1;
wrow=find(P1≠0);
for j=length(wrow):(-1):1
    P(k)=P1(wrow(j));
    k=k+1;
end
P;
```

8.2 附录 B——求解过固定点的最短路径的 Matlab 源程序

```
function [P d]=cn2shorf(W,k1,k2,t1,t2)
[p1 d1]=n2shorf(W,k1,t1);
[p2 d2]=n2shorf(W,t1,t2);
[p3 d3]=n2shorf(W,t2,k2);
dt1=d1+d2+d3;
[p4 d4]=n2shorf(W,k1,t2);
[p5 d5]=n2shorf(W,t2,t1);
[p6 d6]=n2shorf(W,t1,k2);
dt2=d4+d5+d6;
if dt1<dt2
    d=dt1;
    P=[p1 p2(2:length(p2)) p3(2:length(p3))];
else
    d=dt2;
    P=[p4 p5(2:length(p5)) p6(2:length(p6))];
end
P
d
```

8.3 附录 C——绘制无向图的的 Mathematica 源程序

```
a = SparseArray[{{1, 25} -> 1, {1, 2} -> 1, {2, 3} -> 1, {25, 24} ->
1, {25, 26} -> 1, {4, 25} -> 1, {4, 3} -> 1, {4, 5} ->
1, {4, 24} -> 1, {5, 24} -> 1, {5, 6} -> 1, {6, 24} ->
1, {26, 24} -> 1, {6, 7} -> 1, {6, 23} -> 1, {23, 24} ->
1, {23, 22} -> 1, {23, 21} -> 1, {23, 26} -> 1, {26, 27} ->
1, {21, 27} -> 1, {7, 22} -> 1, {7, 8} -> 1, {8, 22} ->
1, {8, 9} -> 1, {22, 21} -> 1, {22, 9} -> 1, {9, 17} ->
1, {9, 21} -> 1, {9, 10} -> 1, {9, 15} -> 1, {9, 16} ->
1, {17, 21} -> 1, {17, 16} -> 1, {18, 17} -> 1, {18, 16} ->
1, {18, 19} -> 1, {18, 20} -> 1, {19, 20} -> 1, {20, 21} ->
1, {10, 11} -> 1, {10, 13} -> 1, {10, 15} -> 1, {15, 13} ->
1, {15, 14} -> 1, {15, 16} -> 1, {11, 12} -> 1, {13, 12} ->
1, {14, 12} -> 1, {14, 16} -> 1, {14, 13} -> 1, {11, 13} ->
1}, {27, 27}]

a = a + Transpose[a]

a_graph =
AdjacencyGraph[a, VertexLabels -> Automatic, DirectedEdges -> False,
VertexLabelStyle -> Directive[20],
VertexStyle -> {1 -> Orange, 12 -> Yellow, 15 -> Green,
27 -> Orange},
```

```
EdgeStyle -> {1 \[UndirectedEdge] 25 -> Red,
25 \[UndirectedEdge] 24 -> Red, 24 \[UndirectedEdge] 23 -> Red,
23 \[UndirectedEdge] 21 -> Red, 21 \[UndirectedEdge] 9 -> Red,
9 \[UndirectedEdge] 15 -> Red, 15 \[UndirectedEdge] 13 -> Red,
13 \[UndirectedEdge] 12 -> Red, 12 \[UndirectedEdge] 11 -> Red,
11 \[UndirectedEdge] 10 -> Red, 10 \[UndirectedEdge] 9 -> Red,
9 \[UndirectedEdge] 21 -> Red, 21 \[UndirectedEdge] 27 -> Red}]


b = SparseArray[{{1, 2} -> 1, {2, 3} -> 1, {3, 4} -> 1, {4, 5} ->
1, {5, 6} -> 1, {6, 7} -> 1, {7, 8} -> 1, {9, 10} ->
1, {10, 11} -> 1, {11, 12} -> 1, {12, 13} -> 1, {13, 14} ->
1, {14, 15} -> 1, {15, 16} -> 1, {17, 18} -> 1, {18, 19} ->
1, {19, 20} -> 1, {20, 21} -> 1, {21, 22} -> 1, {22, 23} ->
1, {23, 24} -> 1, {25, 26} -> 1, {26, 27} -> 1, {27, 28} ->
1, {28, 29} -> 1, {29, 30} -> 1, {30, 31} -> 1, {31, 32} ->
1, {33, 34} -> 1, {34, 35} -> 1, {35, 36} -> 1, {36, 37} ->
1, {37, 38} -> 1, {38, 39} -> 1, {39, 40} -> 1, {41, 42} ->
1, {42, 43} -> 1, {43, 44} -> 1, {44, 45} -> 1, {45, 46} ->
1, {46, 47} -> 1, {47, 48} -> 1, {49, 50} -> 1, {50, 51} ->
1, {51, 52} -> 1, {52, 53} -> 1, {53, 54} -> 1, {54, 55} ->
1, {55, 56} -> 1, {57, 58} -> 1, {58, 59} -> 1, {59, 60} ->
1, {60, 61} -> 1, {61, 62} -> 1, {62, 63} -> 1, {63, 64} ->
1, {1, 9} -> 1, {2, 10} -> 1, {3, 11} -> 1, {4, 12} ->
1, {5, 13} -> 1, {6, 14} -> 1, {7, 15} -> 1, {8, 16} ->
1, {9, 18} -> 1, {10, 19} -> 1, {11, 20} -> 1, {12, 21} ->
1, {13, 22} -> 1, {14, 23} -> 1, {15, 24} -> 1, {17, 25} ->
1, {18, 26} -> 1, {19, 27} -> 1, {20, 28} -> 1, {21, 29} ->
1, {22, 30} -> 1, {23, 31} -> 1, {24, 32} -> 1, {25, 34} ->
1, {26, 35} -> 1, {27, 36} -> 1, {28, 37} -> 1, {29, 38} ->
1, {30, 39} -> 1, {31, 40} -> 1, {33, 41} -> 1, {34, 42} ->
1, {35, 43} -> 1, {36, 44} -> 1, {37, 45} -> 1, {38, 46} ->
1, {39, 47} -> 1, {40, 48} -> 1, {41, 50} -> 1, {42, 51} ->
1, {43, 52} -> 1, {44, 53} -> 1, {45, 54} -> 1, {46, 55} ->
1, {47, 56} -> 1, {49, 57} -> 1, {50, 58} -> 1, {51, 59} ->
1, {52, 60} -> 1, {53, 61} -> 1, {54, 62} -> 1, {55, 63} ->
1, {56, 64} -> 1, {2, 9} -> 1, {3, 10} -> 1, {4, 11} ->
1, {5, 12} -> 1, {6, 13} -> 1, {7, 14} -> 1, {8, 15} ->
1, {9, 17} -> 1, {10, 18} -> 1, {11, 19} -> 1, {12, 20} ->
1, {13, 21} -> 1, {14, 22} -> 1, {15, 23} -> 1, {16, 24} ->
1, {18, 25} -> 1, {19, 26} -> 1, {20, 27} -> 1, {21, 28} ->
1, {22, 29} -> 1, {23, 30} -> 1, {24, 31} -> 1, {25, 33} ->
1, {26, 34} -> 1, {27, 35} -> 1, {28, 36} -> 1, {29, 37} ->
1, {30, 38} -> 1, {31, 39} -> 1, {32, 40} -> 1, {34, 41} ->
1, {35, 42} -> 1, {36, 43} -> 1, {37, 44} -> 1, {38, 45} ->
1, {39, 46} -> 1, {40, 47} -> 1, {41, 49} -> 1, {42, 50} ->
1, {43, 51} -> 1, {44, 52} -> 1, {45, 53} -> 1, {46, 54} ->
1, {47, 55} -> 1, {48, 56} -> 1, {50, 57} -> 1, {51, 58} ->
```

```
1, {52, 59} -> 1, {53, 60} -> 1, {54, 61} -> 1, {55, 62} ->
1, {56, 63} -> 1}, {64, 64}]

b = b + Transpose[b]

b_graph =
AdjacencyGraph[b, VertexLabels -> Automatic, DirectedEdges -> False,
VertexLabelStyle -> Directive[20],
VertexStyle -> {1 -> Orange, 30 -> Yellow, 55 -> Yellow,
39 -> Green, 62 -> Green, 64 -> Orange},
EdgeStyle -> {1 \[UndirectedEdge] 2 -> Red,
2 \[UndirectedEdge] 3 -> Red, 3 \[UndirectedEdge] 4 -> Red,
4 \[UndirectedEdge] 5 -> Red, 5 \[UndirectedEdge] 13 -> Red,
13 \[UndirectedEdge] 22 -> Red, 22 \[UndirectedEdge] 30 -> Red,
30 \[UndirectedEdge] 39 -> Red, 39 \[UndirectedEdge] 47 -> Red,
47 \[UndirectedEdge] 56 -> Red, 56 \[UndirectedEdge] 64 -> Red}]

c = SparseArray[{{1, 2} -> 1, {1, 4} -> 1, {1, 5} -> 1, {2, 3} ->
1, {2, 4} -> 1, {3, 4} -> 1, {3, 8} -> 1, {3, 9} -> 1, {8, 9} ->
1, {9, 10} -> 1, {9, 11} -> 1, {4, 5} -> 1, {4, 6} -> 1, {4, 7} ->
    1, {10, 11} -> 1, {10, 13} -> 1, {7, 11} -> 1, {7, 12} ->
1, {7, 6} -> 1, {5, 6} -> 1, {6, 12} -> 1, {6, 13} ->
1, {11, 12} -> 1, {11, 13} -> 1, {12, 13} -> 1}, {13, 13}]

c = c + Transpose[c]

c_graph =
AdjacencyGraph[c, VertexLabels -> Automatic, DirectedEdges -> False,
VertexLabelStyle -> Directive[20],
VertexStyle -> {1 -> Orange, 9 -> Yellow, 13 -> Orange},
EdgeStyle -> {1 \[UndirectedEdge] 5 -> Red,
5 \[UndirectedEdge] 6 -> Red, 6 \[UndirectedEdge] 13 -> Red}]

d = SparseArray[{{1, 2} -> 1, {2, 3} -> 1, {3, 4} -> 1, {4, 5} ->
1, {6, 7} -> 1, {7, 8} -> 1, {8, 9} -> 1, {9, 10} ->
1, {11, 12} -> 1, {12, 13} -> 1, {13, 14} -> 1, {14, 15} ->
1, {16, 17} -> 1, {17, 18} -> 1, {18, 19} -> 1, {19, 20} ->
1, {21, 22} -> 1, {22, 23} -> 1, {23, 24} -> 1, {24, 25} ->
1, {1, 6} -> 1, {2, 7} -> 1, {3, 8} -> 1, {4, 9} -> 1, {5, 10} ->
1, {6, 11} -> 1, {7, 12} -> 1, {8, 13} -> 1, {9, 14} ->
1, {10, 15} -> 1, {11, 16} -> 1, {12, 17} -> 1, {13, 18} ->
1, {14, 19} -> 1, {15, 20} -> 1, {16, 21} -> 1, {17, 22} ->
1, {18, 23} -> 1, {19, 24} -> 1, {20, 25} -> 1}, {25, 25}]

d = d + Transpose[d]

d_graph =
AdjacencyGraph[d, VertexLabels -> Automatic, DirectedEdges -> False,
```

```
VertexLabelStyle -> Directive[20],
VertexStyle -> {1 -> Orange, 14 -> Green, 18 -> Yellow,
25 -> Orange}]
```

8.4 附录 D-绘制天气概率图的 Python 程序

```
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 13 07:22:40 2020

@author: Administrator
"""
import matplotlib.pyplot as plt
import seaborn as sns
import tikzplotlib
plt.style.use('ggplot')
plt.rcParams['font.sans-serif']=['SimHei'] #显示中文标签
plt.rcParams['axes.unicode_minus']=False   #显示负号
# 第一个图
x = [0,1]
y = [0.5,0.5]
plt.bar(x,y,width=0.4,color='tomato')
plt.xticks(x,fontsize=16)
plt.yticks(y,fontsize=16)
plt.xlim(-0.7,1.7)
plt.ylim(0,0.7)
plt.xlabel('天气分布',size=18)
plt.ylabel('分布概率',size=18)
tikzplotlib.save('C:\picture1.tex')

# 第二个图
x = [0,1,2]
y = [0.4,0.4,0.2]
plt.bar(x,y,width=0.4,color='tomato')
plt.xticks(x,fontsize=16)
plt.yticks(y,fontsize=16)
plt.xlim(-0.5,2.5)
plt.ylim(0,0.55)
plt.xlabel('天气分布',size=18)
plt.ylabel('分布概率',size=18)

tikzplotlib.save('C:\picture4.tex')
```

8.5 附录 E-绘制二维笛卡尔坐标系的 Python 程序

```python
import matplotlib.pyplot as plt
import tikzplotlib
plt.rcParams['font.sans-serif']=['SimHei'] #显示中文标签
plt.rcParams['axes.unicode_minus']=False   #显示负号
x_0 = [i for i in range(1,6)]
x = []
for i in range(1,6):
    x = x+x_0
# x = [i for i in range(0,30,5)]
y_0 = [0,0,0,0,0]
y=[]
for i in range(1,6):
    y = y + [j+i for j in y_0 ]
plt.scatter(x,y,color='tomato')
plt.plot((1,1),(0.75,5.25),linewidth=0.8,color = 'steelblue')
plt.plot((2,2),(0.75,5.25),linewidth=0.8,color = 'steelblue')
plt.plot((3,3),(0.75,5.25),linewidth=0.8,color = 'steelblue')
plt.plot((4,4),(0.75,5.25),linewidth=0.8,color = 'steelblue')
plt.plot((5,5),(0.75,5.25),linewidth=0.8,color = 'steelblue')
plt.plot((0.8,5.2),(1,1),linewidth=0.8,color = 'steelblue')
plt.plot((0.8,5.2),(2,2),linewidth=0.8,color = 'steelblue')
plt.plot((0.8,5.2),(3,3),linewidth=0.8,color = 'steelblue')
plt.plot((0.8,5.2),(4,4),linewidth=0.8,color = 'steelblue')
plt.plot((0.8,5.2),(5,5),linewidth=0.8,color = 'steelblue')
```