Final Examination Report

School of Software, Yunnan University

Mathematical Foundation of Data Science

Student id	Name	Score
20183290375	MD MAHFUZUR RAHMAN（毒液）	
20183290716	SAMEER ZULKAR SYED（陈也军）	
20183290502	MD HASAN ALI SHEIKH（哈桑）	
20183290404	MD ABDUL WADUD PRINCE(吴磊)	
20183290253	AL HASIB(阿布)	
20183290541	MD APEL MAHMUD(马瑞)	

Semester:___________________________2021 Fall

Group Leader: ___MD MAHFUZUR RAHMAN

Group Leader(ID):_______________20183290375

Email:________________rony@mail.ynu.edu.cn

Submission Time: ________________12.01.2022

Work Division

No.	Student number	Name	Major	Division
1	20183290375	MD MAHFUZUR RAHMAN(毒液)	Software Engineering	
2	20183290716	SAMEER ZULKAR SYED(陈也军)	Software Engineering	
3	20183290502	MD HASAN ALI SHEIKH(哈桑)	Software Engineering	
4	20183290404	MD ABDUL WADUD PRINCE(吴磊)	Software Engineering	
5	20183290253	AL HASIB(阿布)	Software Engineering	
	20183290541	MD APEL MAHMUD(马瑞)	Software Engineering	

The inspection rules

Major: **Software Engineering** Grade: **2018** Professor:

Course: **Mathematical Foundation of Data Science** Project Name: **Crossing the Desert**

Item	Score	Evaluation Criterion				Final Score
Feasibility	15	Feasible and Innovative	Feasible	Partly feasible	Infeasible	
Achieve the desired goal	15	Fully achieved	Basically reached	Unforeseen	Not reached	
Team cooperation	15	Excellent	Good	Average	Bad	
Project difficulty	30	Very difficult	More difficult	Moderate difficulty	Easy	
Document quality	30	Clear structure , rich content	rich content	Clear structure	incomplete	
Total Score						
Comment:						
Signature of the professor						

Content

A crossing the Desert

Summary

Aiming at the game target of crossing desert game, consider whether the weather is known all the way, whether there is a multi-player game, etc., establish a model, give strategies, and solve for specific levels.

In Question One, the weather conditions are known throughout the journey. The desert map is abstracted and simplified, and the key path is calculated by the shortest-circuit algorithm to obtain a directional map. On this basis, a dynamic planning model is constructed, and the optimal strategy in a given initial state is solved by using the memory search implementation algorithm. Through the multi-search algorithm, all the initial states are searched to obtain the global optimal strategy of the specified level. In the optimal strategy of the first and second levels, the final remaining funds are 10470 yuan and 12730 yuan, respectively.

In question two, because of the uncertainty of the environment, the cumulative effect of uncertainty will affect the final decision, so we have established a simplified model based on key nodes, and thus established the user's decision path. At the same time, in order to ensure the player's earnings and best deal with uncertain environment, players should adopt some specific strategies, such as because the next day can still be hot, so in hot weather should also walk as usual. Finally, the planning model can be established with the remaining funds as the target function, time and resources as the constraints, and the connection of parameters can be established, so as to analyze the basic analysis strategies that the user should adopt, such as going to the end point when only a certain amount of resources or time remains, analyzing the relationship between revenue and cost to decide whether ore should be mined, etc. Based on this model, the final general strategy can be obtained after analyzing the status and cost and benefit of the key node location of question three or four.

In question three, there is a multi-player game. In view of making all the decisions at the starting point, a complete information static game model is established. After simplifying the map, the decision set is calculated using the dynamic planning model, the winning matrix is obtained, and 2 Nash equilibriums are obtained. Then we establish a linear planning model and find out the hybrid strategy. For each step to make a decision and the weather is unknown, establish the Markov decision-making process, introduce the degree of risk acceptance, through the Monte Carlo simulation analysis of the impact of risk acceptance on decision-making and the final remaining funds, to arrive at a general strategy.

Keywords: Dynamic planning, Static game, Shortest path, Desert game.

First, overview

1.1 Question Background

In the Desert Crossing game, players use a map to buy a certain amount of water and food with initial funds and walk through the desert from the starting point. Different weather conditions can be encountered along the way, and additional funds or resources can be replenished in mines and villages, with the goal of reaching the end within a specified time frame and retaining as much money as possible.

the case of limited time, resources and funds, it is necessary to consider whether the environment is determined, the game between players, to make reasonable decisions for different situations, so as to maximize the benefits of the game without failure.

For the determination of the environment, can be divided into the whole weather conditions known, only know the weather conditions of the day two situations, respectively, discuss the best strategy and the best strategy in both cases. For the game between players, it is necessary to discuss what strategies players should adopt under multi-player game conditions.

Based on the key nodes that the player may make decisions, this paper discusses the decision-making process of dynamic planning, the stage decision path in uncertain state, and the game under multi-player conditions, and gradually solves the problems in this background after simplifying the map through the network model.

1.2 Target task

Question 1: There must be a global optimization strategy for a given map when the weather conditions are known throughout the course and only one player is available. Consider the rules of the game, establish a mathematical model that can be extended to the general situation, and solve the global optimal strategy for the first and second levels.

Question 2: How to make the optimal decision in the uncertain state of the environment, so as to maximize profits. Consider key nodes and different decision stages to establish decision paths that allow players to do so without failing and maximizing when they only know the weather of the day.

Question 3: How to make a strategy in advance or make the best strategy based on the environment when multiple people are involved and decisions are influenced by each other. Consider making all the strategies in advance and dynamically making the strategy, establish the corresponding decision model, and get the general best strategy of the game.

Second, Symbol Description and Model Foundation

2.1 Symbol Description

The model used in this article, as shown in Table 1, is mainly named after the alphabetical label pattern, and some of the less used variables are defined when used.

Table 1 symbols and meaning

Symbol	**Meaning**	**Symbol**	**Meaning**	**Symbol**	**Meaning**
$cost$	spend	m_{init}	Initial capital	w_{cost}	Benchmark price of water
Cap_{basic}	Basic income	C_{total}	Total cost	f_{cost}	Base price of food
w_{sun}	Basic water	w_{hot}	High temperature basic water consumption	w_{dust}	Basic water consumption of sandstorm
f_{sun}	Consumption on sunny days	f_{hot}	High temperature basic food consumption	f_{dust}	Basic food consumption in sandstorms
m_{left}	Clear basic food consumption	f_{left}	Surplus food	w_{left}	Remaining water
f_{buy}	Remaining money	w_{buy}	Purchased water	d_{total}	Total days
d_{cur}	Food purchased	$Point$	Player position	K	Number of stages
S	Current day State	X	Decision making	D	Decision set
d_{min}	Detour the mine to the end	cap_{total}	All benefits	$cost_{total}$	All spent
c_{sun}	Clear basic consumption	C_{hot}	Hot basic consumption	C_{dust}	Basic consumption of sand and dust

2.2 The network Model is simplified

In Questions 1 and 2, there is only one player and there is no game problem between players. Also, because all areas of the desert have the same weather, except for the starting point, end point, village, mine, and key decision node, the player only needs to focus on the starting point, end point, village, mine area, and some key nodes where the player may make different decisions. At the same time, considering the game goal, you can delete some paths that do not affect the calculation of optimal solutions. From this, the original desert map can be abstracted and simplified into a directional map. As shown in the following image, the numbers on the arrows represent the shortest distance between the two nodes, where the player can decide to wait or advance.

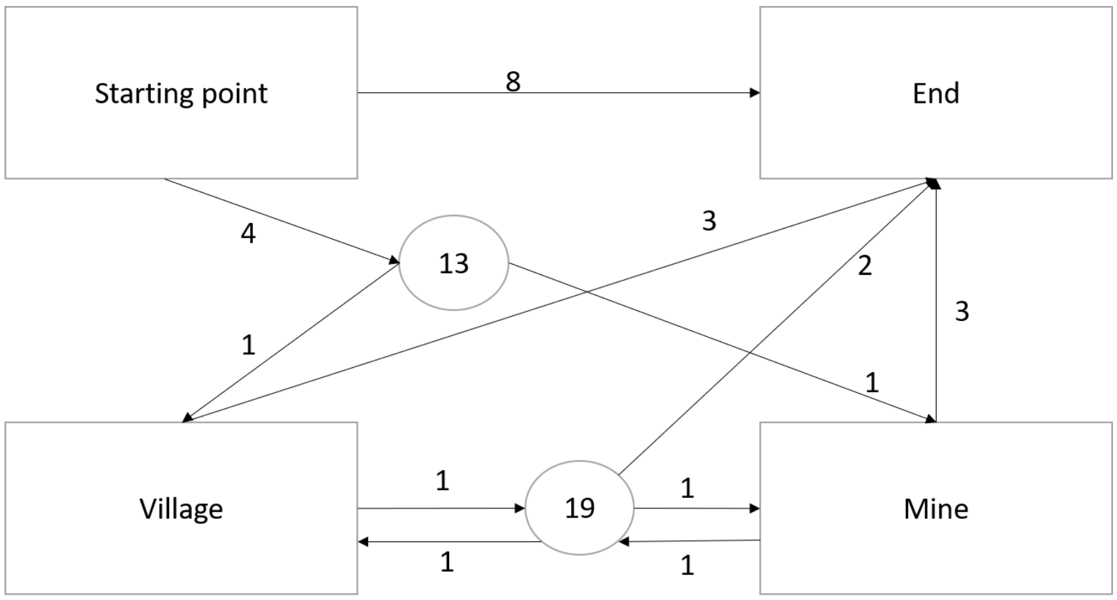


Figure 1 The network model simplifies the indication

In question three/four, because the state is uncertain, there is a path decision problem, the effect of the point is not equivalent, need to find the place with the most cumulative state. We used the adjusted *Dijkstra* algorithm and matching algorithm to simplify the model, get all the paths between points, through the matching algorithm, according to the actual situation to calculate the overlap in the path, and select some nodes as the key nodes. For example, in the fourth level, the player can reach node 13 and then decide whether to go to the mine or village, 13 is the key node, the fourth level simplified network model as shown in Figure 1. Refer to Appendix C for specific algorithms.

Third, the best strategy under known conditions of the whole weather conditions

3.1 Analysis

In this question, the weather conditions are known in advance for each day throughout the game window, and the game has a limited time limit, i.e. the game ends in a limited step. Therefore, the total number of player action strategies is limited, and there must be an optimal strategy. For problems with a limited total number of policies, the optimal strategy can be found by exhaustive method. However, there are many game strategies in this problem, and the calculation of simply using the exhaustive method is too large. Using dynamic planning method, the decision sequence (overall strategy) in multi-stage decision-making problems can be transformed into several sub-strategies, which can effectively reduce the amount of computation.

3.2 Model building

3.2.1 Vector model

According to the method described in section 2.2, the maps of the first and second levels in this question can be reduced to the directional diagram shown in Figure 2, respectively.

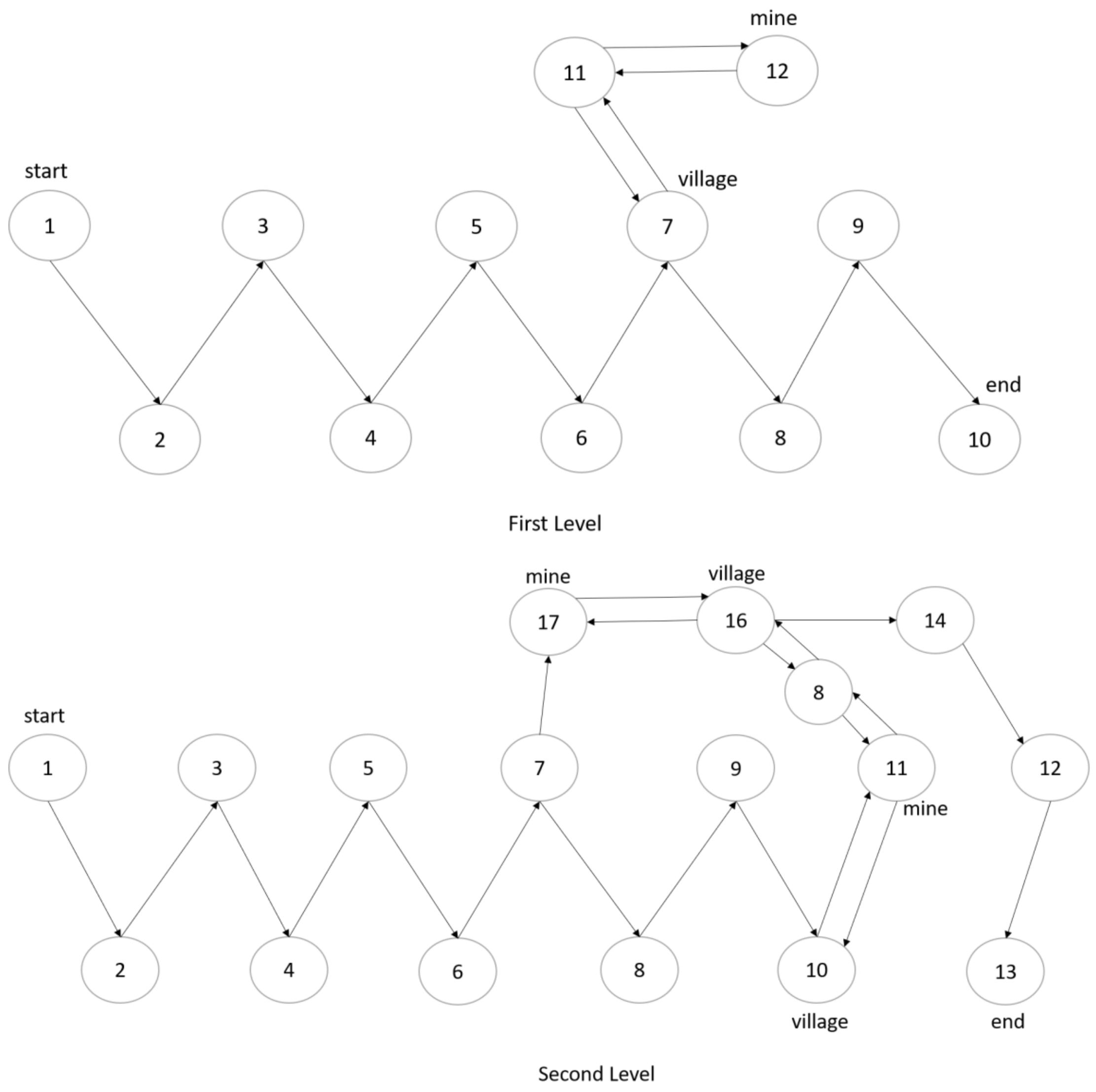


Figure 2 First pass Second level 2 First level, second level: there is a directional model

3.2.2 Dynamic planning model

(1) Stage variables

Considering the "days as the basic time unit" in the rules of the game, select "days" as the basis for the segmentation of dynamic planning, the number of stages of dynamic planning is equal to the current number of days in the game, that is

$$k = d_{cur} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

(2) The state variable

The state of Stage k is represented by s_k, determined by the player's position and the amount of resources (water and food) he or she carries.

$$s_k = g(point, w_{left}, f_{left}) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

s_k satisfies the aftereffects, i.e. once the s_k is determined, the best strategy from that stage to the end of the game (i.e. the best sub-strategy from s_k) is not affected by the state of the previous stages of stage k. Once the status of each stage is determined, the entire game is completely determined.

(3) Behavioral decision-making

The decision is represented by x_k (s_k) when the player in the s_k state in stage k, and Dk(s_k) represents the set of decisions allowed when the player in the s_k state in stage k, so $x_k (s_k) \in D_k (s_k)$. Depending on the rules of the game, you can enumerate the set of decisions that players make when they are in different states. Taking the state of the player in the mine (point 12) in the first pass as an example, the decision set can be represented as follows:

Table 2 A set of decisions made when a player is in a mine in the first pass

Decision number	Weather	Path	Weather to mine	Decision number	Weather	Path	Weather to mine
A_1	Sunny	Move to point 11	No	E	High temperature	Stay	Yes
B	Sunny	Stay	Yes	F	High temperature	Stay	No
C	Sunny	Stay	No	G	Sandstorm	Stay	Yes
D_1	High temperature	Move to point 11	No	H	Sandstorm	Stay	No

In dynamic planning, a specific set of decisions can be calculated for each stage and state.

(4) State transfer function

At each stage, players may change resources and positions, resulting in state changes. The state transfer function is shown in formula (3)-(5).

$$w_{left} = \begin{cases} w_{left} - n * w_{sun} & s_k \in \text{sunny} \\ w_{left} - n * w_{hot} & s_k \in \text{High temperature} \\ w_{left} - w_{dust} & s_k \in \text{sandstorm} \end{cases} \qquad \text{.............(3)}$$

$$f_{left} = \begin{cases} f_{left} - n * f_{sun} & s_k \in \text{sunny} \\ f_{left} - n * f_{hot} & s_k \in \text{High temperature} \\ f_{left} - f_{dust} & s_k \in \text{sandstorm} \end{cases} \qquad \text{.............(4)}$$

$$s_{k+1} = g(point_{k+1}, w_{left}, f_{left}) \qquad \text{.............(5)}$$

In (3)-(5),

$$n = \begin{cases} 3 & s_k \in \text{mining} \\ 2 & s_k \in \text{move} \\ 1 & s_k \in \text{stay} \cap \text{No mining} \end{cases} \qquad \text{...............(6)}$$

(5) Stage profit and loss function

The goal of the game is to retain as much money as possible when the end point is reached, and to use the change of funds as a stage profit and loss function.

$$cost(s_k,x_k) = -p * w_{cost} * w_{buy} - p * f_{cost} * f_{buy} + q * cap_{basic} \quad \text{................(7)}$$

$$p = \begin{cases} 2 & s_k \in \text{Supply \& k not equal 0} \\ 1 & k = 0 \\ 0 & else \end{cases} \quad \text{...............(8)}$$

Based on the above instructions, the dynamic planning model can be summarized as:

$$\begin{cases} cost_{total}(s_0) = m_{init} - m_{left}(s_0) \\ cost_{total}(s_{k+1}) = cost(s_k, x_k) + cost_{total}(s_k) \quad k = 0, 1, 2, ..., d_{total} - 1 \end{cases} \quad \text{....(9)(10)}$$

Where $x_k \in D_k(s_k)$, $m_{left}(s_0)$ is the remaining funds after day 0 replenishment.

Traversing all the states at the end of the last phase, with the least total cost (the most money remaining), is the optimal strategy in that initial state. In the model solution, all initial states of a specific level are considered to obtain the global optimal strategy for that level.

3.3 Model solution

3.3.1 Algorithm

Because there are different replenishment strategies at the starting point (day 0), these strategies affect the initial state, which in turn affects the decision sequence and final outcome of the game. Therefore, using the multi-search algorithm, you consider day 0 water and food replenishment, and search for all replenishment strategies for day 0, each with an initial state. Once the initial state is determined, the optimal strategy for each initial state is calculated using the dynamic planning model. Finally, the optimal policy is selected from the optimal strategy corresponding to all initial states, that is, the global optimal policy for the level.

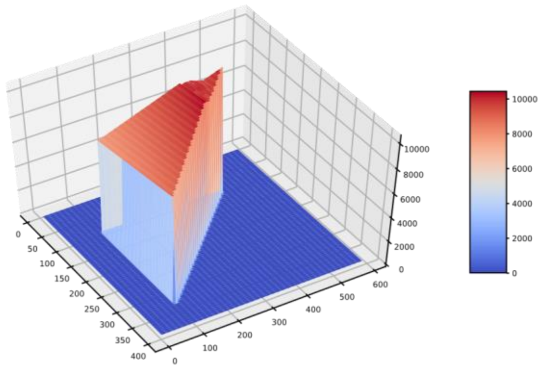


Figure 3 Level 1: The final remaining funds for optimal strategies under different initial replenishment conditions

In maps with villages, too many strategies for resupplying villages greatly increase computational complexity. Therefore, the algorithm is optimized with a "use first, book later" strategy: allowing a gap in the resource, and checking if the gap can be met the last time the resource passes through the village or reaches the end point. If it can be met, fill the gap, deduct the funds, and ensure a minimum supply, and if it cannot be met (overweight or underfunded the last time you passed through the village), prune and stop the continued push of this state. In addition, considering that the solution direction of the model and the direction of the state transfer function are both forward, the classical dynamic planning algorithm is improved by combining the memory search algorithm.

The algorithm is implemented using Python, and the source code is in Appendix B.

3.3.2 Level 1

After multiple searches, the final remaining funds for the optimal strategy under different initial replenishment conditions are shown in Figure 3. The X and Y coordinates at the bottom of the image represent the amount of water and food purchased on day 0, respectively, and the function value is the final remaining funds. Areas with a final fund of 0 (dark blue) indicate that the number of resources under these initial replenishment conditions exceeds the load limit or is not sufficient to support the player to reach the end point.

It has been calculated that the optimal strategy for the first level is to leave on day 0 with 178 cases of water and 333 boxes of food, to reach the village on the 8th day and resupply 163 boxes of water, to arrive at the mine on the 10th day, and to stay at the mine on the 11th to 19th day, where No mining on day 11, 17, mining on other dates, another 21st day to the village and replenish 36 boxes of water and 16 boxes of food; 24th day to the end, water and food are just exhausted; and the final remaining funds 10470 yuan. The detailed calculations are provided in the appendix.

3.3.3 Level 2

The specific solution process is similar to the first level and is not repeated. The optimal strategy for the second pass is calculated to depart on day 0 with 130 cases of water and 405 boxes of food, to reach the village at point 16 (area 39 in the original map) on day 10 and to resupply 10 cases of water on day 11 Stay in the village and resupply 179 boxes of water, arrive at the mine at point 17 (area 30 in the original map) on day 12, dig at the mine on day 13 to 18, and reach the village at point 16 on day 19 and fill it up

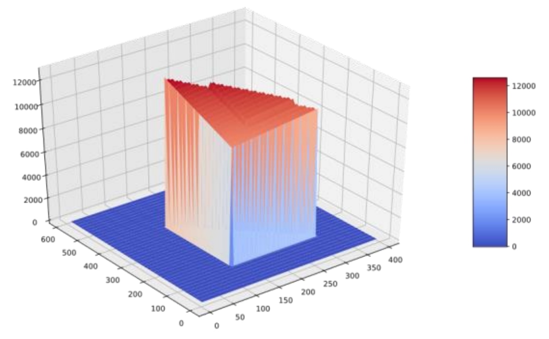


Figure 4 Level 2: The final remaining funds for optimal strategies under different initial replenishment conditions

Give 196 boxes of water and 86 boxes of food, arrive at the mine at point 11 (area 55 in the original map) on day 21, mine at mine on day 22-28, reach the finish line on day 30, water and food are just running out, and the final remaining funds are $12,730. The detailed calculations are provided in the appendix.

Fourth, the weather is unknown in general the best strategy

4.1 Problem Analysis

The main constraint of the player's best strategy when only the weather is known that day is to maximize the game's revenue while ensuring that the game does not fail. In order not to guarantee the failure of the game, the player must reach the end point within a specified time, while ensuring that there is still water and food left, which requires the player to reasonably arrange the purchase of items and the time of mining.

The most important factors influencing decision-making are the ratio of sandstorms to hot weather and the availability of villages. The probability of sandstorms is less than 100%, otherwise the player will not be able to move forward, the probability of hot weather is 0%-100%, i.e. it may be all hot or sunny along the way, the village may or may not have, and the mine is preset in this problem, otherwise there is no need to study how to maximize the benefits.

Next, we will then simplify the model, and then analyze the impact of decision-making factors, considering the uncertainty of some factors, some factors to determine the situation, the general player's optimal decision.

4.2 Modeling

4.2.1 The network Model is simplified

Based on the derivation of 2.2 and the algorithm of Appendix A, we can presumptive the network to a model with only key nodes. Take the fourth level as an example, players need to go from the starting point to "mine" or "end point", in order to achieve the optimal strategy, to avoid walking caused more consumption, the shortest path will inevitably be selected on the way, most of the positions on the way is equivalent. But at the 13/19 position, the player can decide whether to go to the mine or village, continue mining, or go to the end point, which is also the key node.

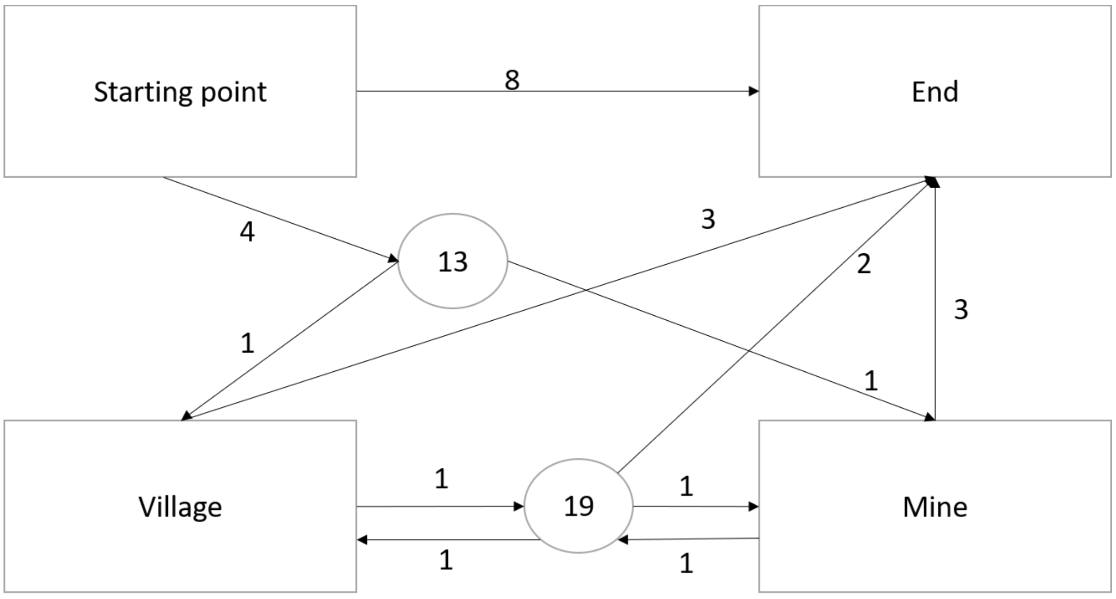


Figure 5 Level 4 - Model Simplification

In this diagram, the numbers on the arrows represent blocks that need to be walked between two locations, where players can choose to walk or stop in place. At the same time, players can choose to stop not working or "mine" when they arrive at mine. Since the player can't buy it back to Start and goes to End, there is no straight line from Mine to Start or End to another location.

4.2.2 Decision Path Establishment

After establishing a simplified version of the network model, we can establish the player's general decision path, including the various stages of decision-making and their respective trigger conditions, and then quantify the effect of comparing different paths, so as to obtain the optimal strategy. The decision-making road is divided into two directions:

(1) Go directly to the end point, including the direct termination stage:
(2) Detour to the mine, and then point, including the initial stage, mining stage and termination stage.

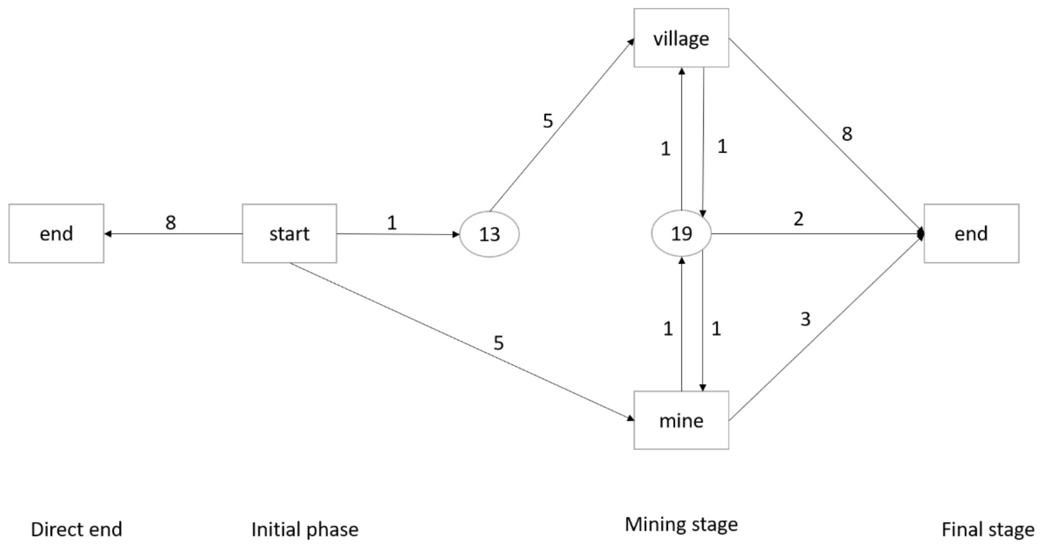


Figure 6 Level 4 - Decision path

Also take the fourth level as an example, Figure 6 is a simplified decision model in Figure 5, where players can choose to go directly to the end point, terminate the game, or go to the mine first, and then work or rest, triggering the termination condition when certain conditions are met, such as insufficient water or food, to the end.

Based on this model, we step-by-step analyze the player's optimal decision-making process, i.e., how the player should choose "direct termination" (left) or go to a mine or village (right) according to the conditions, how the player should adopt different strategies according to the weather during the initial stage, mining stage, termination phase, and what conditions will trigger the end of the mining phase and start the termination phase.

4.2.3 Based on influencer strategy analysis

(1) Village impact

If there is no village, the player needs to arrange the time to the mine and the initial purchase of food, while ensuring that the end point can be reached at the expected time, and if there is a village, the player needs to arrange the time between the mine and the village to ensure that the village can be reached within the scheduled time to maximize the benefits.

Basic Strategy 1 When there is no village, the player's decision path is to go directly from the starting point to the end point, or from the starting point directly to the mine and then to the end point;

(2) The effect of high temperature

Hot weather is a common desert weather, with water and food consumption 2-3 times higher than normal, and greater consumption during walking or mining. At the same time, because the future weather conditions are unknown, if the hot weather does not go on the day, the future may still be high temperature, so in the non-mine location, players should regardless of whether the high temperature directly to the destination. Decisions made at mine locations should depend on factors such as resource costs, weather, village and distance.

Basic Strategy 2 During the non-mining phase, players should go directly to their destination regardless of the high temperature, and during the mining phase, the benefits and costs should be quantified.

(3) The effects of sandstorms

The effects of sandstorms are similar to those of high temperatures, with the main difference that the underlying consumption of sandstorms is higher than in hot weather, while the ability to mine (and consume three times as much food and water) in sandstorms and the inability to walk in sandstorms. Sandstorm weather can only stay in place or mine, so sandstorms have two main effects: the need to travel to the finish line early to ensure that the end point is reached on time, the need to make decisions about whether to mine according to cost and profit.

Basic Strategy 3 If there is sandstorm weather, if the player chooses to mine, they should go to the end point early, and whether mining in sandstorm weather depends on whether the actual return is higher than the expenditure.

4.2.4 Decision path and stage Analysis

In the previous section we discussed the qualitative effects of several factors, and in this section, we discuss how players should make decisions about paths and decisions that should be made at different stages, including whether to mine or terminate, what weather should mine, whether to go to villages, whether to go to the end, and so on, which ultimately maximizes the remaining money.

The goal of the problem is to maximize the final remaining funds m_{left} given the initial funds m_{init}. We define the number of sandstorm days that may occur as d_{dust}, the total distance between bypass mining and the village detour, the total number of days of mining is d_{mine}, the total subsequent cost $cost_{total}$, the mining revenue cap_{total}, the start-to-end distance dis, and other parameters follow the previous, optimization objectives can be described as:

$$\max m_{left} = m_{init} - cost_{total} + cap_{total}, \quad (11)$$

$$\text{s.t. } cost_{total} = (w_{buy} + w_{village} * 2) * w_{cost} + (f_{buy} + f_{village} * 2) * f_{cost}, \quad (12)$$

$$cap_{total} = d_{mine} * cap_{basic}, \quad (13)$$

$$d_{total} \geq dis + detour + d_{dust} + d_{mine}, \quad (14)$$

$$w_{buy} + w_{village} \geq w_{hot} * (d_{mine} * 3 + dis_{cur} * 2 + detour * 2) + w_{dust} * d_{dust}, \quad (15)$$

$$f_{buy} + f_{village} \geq f_{hot} * (d_{mine} * 3 + dis_{cur} * 2 + detour * 2) + f_{dust} * d_{dust} \quad (16)$$

We can see that the final benefit is determined by the cost of purchasing resources at the starting point and in the village (12), as well as the total proceeds from mining (13), while time (14) and resources (15-16) both need to meet the constraints of not failing the game, and based on this objective function, we can summarize four strategies.

Path Selection 1 If the proceeds from going to the mine may be enough to cover the cost of bypassing and the cost of water and food consumed when the mine cannot be mined, you should not go to the mine.

Path Selection 2 Where there are villages, whether or not to mine depends on the purchase of water and food, the combined cost of going to the village and mine, and, if the cost is cost-effective, refer to the remaining days to purchase water and food.

Termination Condition 1 If you are carrying resources that are just in extreme weather and cannot be purchased additionally, you will need to travel to the end point immediately.

Termination Condition 2 In the event of sand and dust, the end point needs to be made 1-2 days in advance, depending on the distance and the time remaining.

Path selection 1-2 means that mining is cost-effective only if the bypass mining generates positive benefits and the benefits can cover the cost of the bypass, and it is only cost-effective to go to the village if the bypass to the village allows for more revenue to be generated, and the cost of the bypass to the village and the cost of the purchase of resources.

Termination condition 1-2 indicates the need to ensure that the game ends successfully on time. This strategy should be used in specific discussions to consider the path and strategy that should ultimately be taken by comparing potential benefits, remaining resources, and other factors.

4.3 Model solution

4.3.1 Level 3

According to the algorithm of 2.2, it can be found that there is a common position 4 in the shortest path of the player to mine and end point in the third level, that is, the key node, and the simplified decision path is shown in Figure 7. The main consideration is whether the player should go to mining after reaching position 4, what weather should mine, and what should end the mining.

(1) Cost analysis

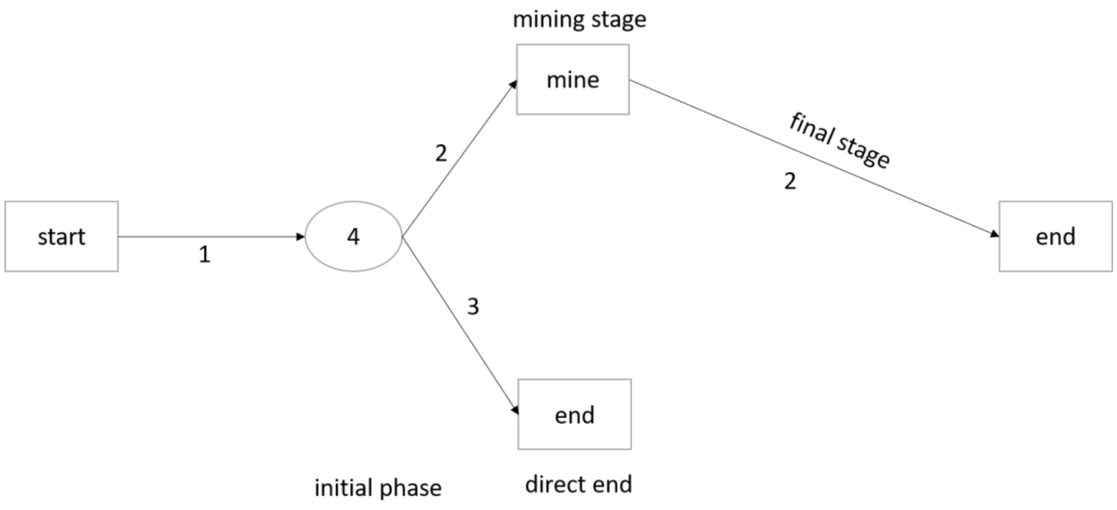


Figure 7 Level 3 - Decision Path

The underlying parameters of the level, if any, are: cap_{basic} = 200 yuan; Mining detour distance: $detour$ = 1 blocks; Sunny base cost: c_s = 125 元; High temperature base cost: c_{hot} = 315 元; Total number of days d_{total} = 10 天; The shortest time to bypass the mine to the end d_{min} = 5 天.

The highest cost of bypass $detour * c_{hot} * 2 = 630$ 元; Revenue from mining on sunny days: $cap_{basic} - 3 * cost_s = -175$ 元; Revenue from mining on hot days: $cap_{basic} - 3 * cost_h = -745$ 元; Number of days to mine: $d_{total} - d_{min} = 5$ 天。

(2) Decision path Analysis

According to the information in the previous section, it is more obvious that mining for the purpose of making money is a loss, so it should be Resources ready to go directly to the end point. water: $w_{hot} * 2 * d_{min} = 216kg$, grain: $f_{hot} * 2 * d_{min} = 144kg$。

Players need to make a decision for the first time when they reach 4, and if day 1 is sunny, it's a water saver $(w_{hot} - w_{sun}) * 2 = 36kg$, Save food $(f_{hot} - f_{sun}) * 2 = 20kg$。

If it is still sunny after reaching 4, it is equivalent to saving a total of 73kg of water and 40kg of food, so detour is 1, so you can choose to bypass to the mine, i.e. go to 3 locations, because even if the day is hot, it can meet the demand of walking water 54kg and grain 36kg on hot days.

When the player arrives at 3, if there is another sunny day, walk to the mine relative to the equivalent of saving 36kg of water and 20kg of grain again, and after arriving at mine 9, if there is another sunny day, since the water and food saved the previous day is sufficient to mine the full consumption on a sunny day, so they should dig in site and earn a base income of 200 yuan.

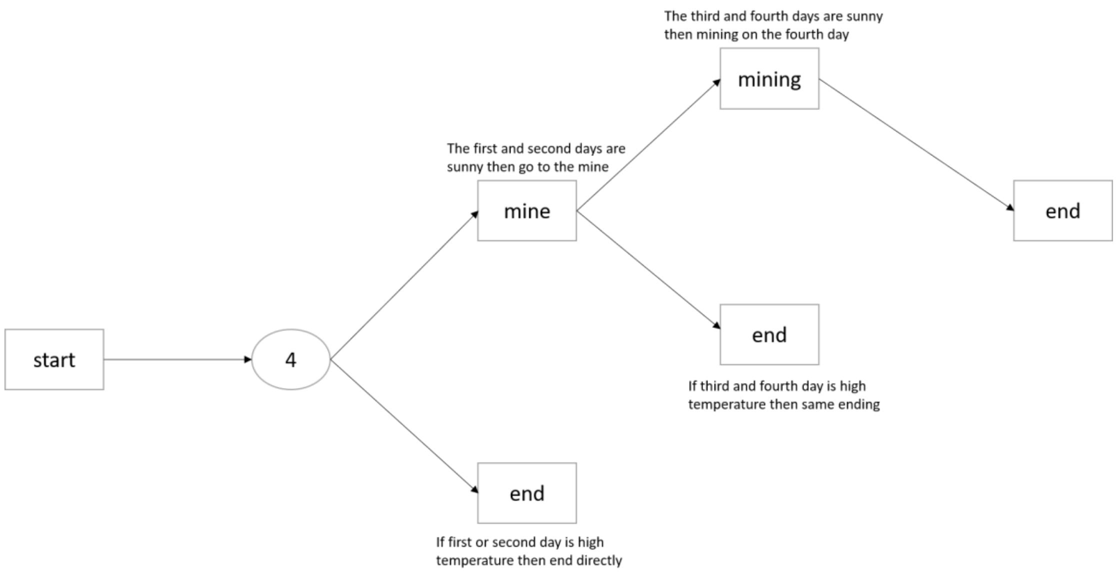

The third and fourth days are sunny then mining on the fourth day
mining
The first and second days are sunny then go to the mine
mine
end
start
4
end
If third and fourth day is high temperature then same ending
end
If first or second day is high temperature then end directly

Figure 8 Level 3 - Decision path selection

(3) Policy Summary

The decision path calculated in the previous section presents the result that food and water prepared for a 4-day high-temperature walk will yield $200 if the fourth day is all sunny.

Table 3 Best strategy for the third level

Initial purchase	$4 * w_{hot} = 216kg$ water and $4 * f_{hot} = 144kg$ food， spend 2520 yuan
Basic strategy	Go to position 4 on Day 1, follow 4-6-13 to the end if the next day is hot, and follow 4-3-9-11-13 to the end if the next day is not hot, where mining is mining early on the fourth day if the 3/4th day is all sunny.
The remaining funds	if there is no mining, remains $m_{init} - 216kg * w_{cost} - 144kg * f_{cost} = 7480$ yuan； If you can mine， remains 7480+200=7680 yuan.

4.3.2 Level 4

According to the above analysis, the fourth level decision path is as follows, after quantifying the cost of mining in the mine, to determine what decisions should be made at several core decision nodes, and to calculate when to go to the end point.

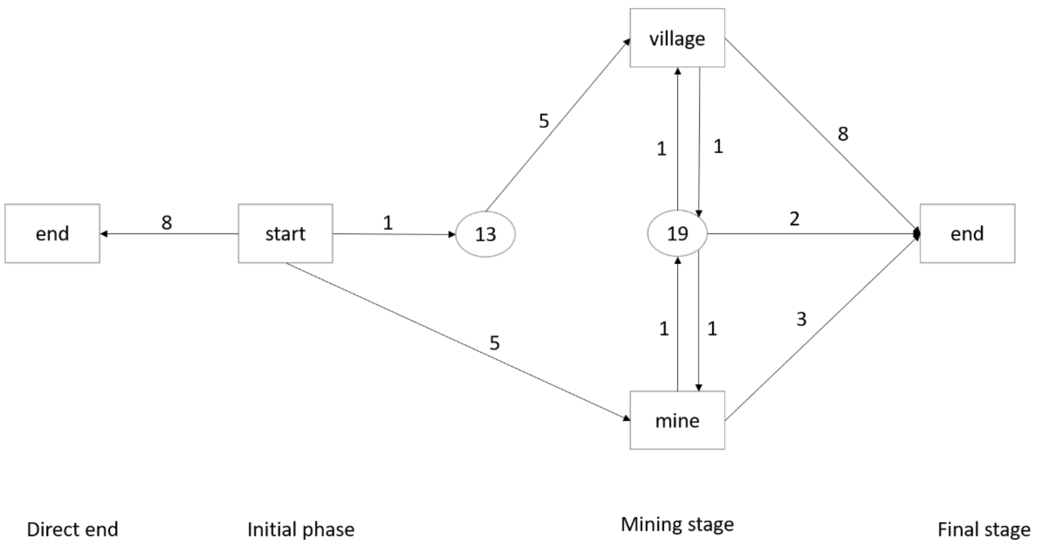


Figure 9 Level 4 - Decision-making process

(1) Cost Analysis

If the basic parameters of the pass are available, the underlying benefits:

$cap_{basic} = 1000$ 元； Mining detour distance： $detour = 0$ blocks； clear weather base consumption $c_{sun} = 125$ 元， basic consumption in hot water $c_{hot} = 315$ 元， sandstorm weather basic consumption $c_{hot} = 350$ 元

In terms of income, as the price of goods purchased in villages doubles, the income from mining should be divided into resources purchased from the starting point before and after the use of resources. Before the use of resources, consistent with the results of the third level, sunny day mining earns 635-yuan, high temperature mining earns 55 yuan, while sandstorm mining loss of 50 yuan; $cap_{basic} - c_{sun} *2*3 = 370$ 元， high-temperature mining gains $cap_{basic} - c_{hot} *2*3 = -890$ 元， the proceeds from sandstorm mining are $cap_{basic} - c_{dust} * 2 * 3 = -1100$ 元。

(2) Basic Analysis of Decision Path

When mining with the resources purchased from the starting point, the net benefits from sunny and high temperatures are 635 yuan and 55 yuan, respectively, although the sandstorm weather loss of 50 yuan, but the subject condition is that the weather is relatively small, so mining in the mine we can be considered to be able to obtain income. In the worst-case scenario, 1200kg is enough to start to the end and have the remaining resources to mine. At the same time, the shortest path from start to finish is to go directly through the mine, so players should buy enough resources at the starting point to mine directly as they pass by for the benefit.

But after running out of resources, the loss of hot weather nearly doubles the profit on sunny days, while the process from mine to village needs to consume a lot of resources, players only know the weather of the day, do not know whether the future is high temperature, so it is not cost-effective to use the village's resources to dig. It should be considered, however, that it may be cost-effective to go to villages to buy resources because of the inconsistency between sunny weather and the consumption of food and water in hot weather.

In summary, players should generally choose the shortest path directly to the mine mining, and should not use village resources mining, the use of unequal resources discussed in the next section.

(3) Purchase strategy and Decision path

The purchase of resources at the starting point should be 1200kg, taking into account the number of sandstorms and high temperature consumption tanks: food s1:1, according to this ratio, just water purchase 720kg, food purchase 480kg, can also be purchased in proportion to the consumption of sunny weather, follow-up analysis.

If it's all hot, the final resources will be consumed proportionally, but if all that was previously encountered before position 13 or 19 is sunny, the ratio of water to food will be unbalanced, which may result in other decision-making methods. But players should not go to the village to buy food, whether or not there is enough water to spare. When the player's food is all purchased in the village, the basic cost of mining is $f_{cost} * 2 * f_{hot} * 3 = 1080$ yuan, i.e. even if you don't need to buy food, mining with food purchased by the village is lost in hot weather.

When food doesn't need to be purchased and water needs to be purchased in the village, the basic cost of high-temperature mining is $w_{cost}*2*w_{hot}*3 = 810$ yuan, and buying water to mine in the village can pay off, but players need to get around the purchase Rows and additional costs, such as going from the 19th to the 14th position, for example, after buying water on the 14th and traveling, need to go to the mine to mine to make up for the consumption, if continuous high temperatures and sandstorm weather, after a simple calculation, the player cannot get additional benefits.

In summary, no matter how the weather changes, you should not go to the village to buy water and food. The final specific decision path and best strategy are:

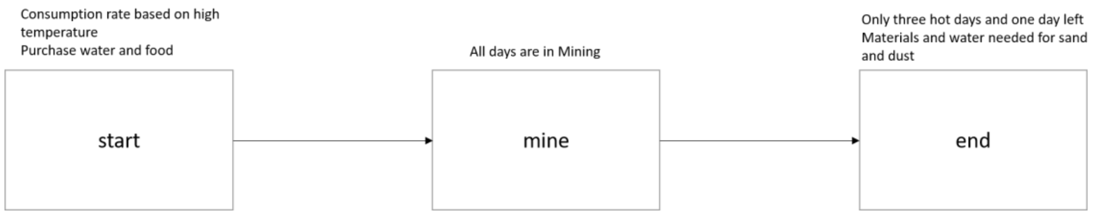

Consumption rate based on high temperature
Purchase water and food
start
All days are in Mining
mine
Only three hot days and one day left
Materials and water needed for sand and dust
end

Figure 10 Level 4 - Decision path selection

4 The best strategy for the fourth level

Initial purchase	720kg of water and 480kg of food, or in proportion to consumption in clear weather
Basic strategy	Go directly to the mine, keep moving forward in addition to sandstorms, and choose to mine no matter what the weather is like when you arrive at the mine
End condition	If the remaining water during mining will be less than 192kg or less than 128kg, choose to follow the shortest path to the end.

Five, multi-player game under the conditions of the strategy

5.1 Problem analysis

In this problem, there are many decision makers in the game, each decision maker maintains its own decision variables and objective functions, and the behavior of decision makers affect each other, so we use the game model to study.

At level 5, the number of players is 2, each player has full information on day 0, and each player's course of action needs to be determined on day 0 and cannot be changed thereafter, resulting in a completely static game of information. Under the analysis framework of static model, we need to determine the game's people, decision sets, utility functions, and winning matrix, so as to calculate the game's Nash equilibrium and formulate the optimal game strategy.

In the sixth level, the number of players is 3, and each player knows the action plan and the number of resources left for the rest of the player's day after the action ends, and then determines their action plan for the next day. Similar to level five, each player wants to take the optimal path from the starting point to the end. But unlike the fifth level, the sixth level players have to make a decision every step of the way, that is, according to each day's environment to re-plan a route that is most beneficial to themselves, that is, can be abstracted for the player to interact with the environment and reaction, can be analyzed using Markov's decision-making process theory tools.

5.2 The fifth pass Model is Established and Solved

5.2.1 Modeling

First of all, we set up a complete information static game model for the fifth level. (1) People in the bureau.

In a response, the participants who have the right to decide on their own course of action are referred to as the in-office persons. In this model, the person is the player. We use I to represent a collection of people in the bureau.

2) Decision collection In a game of countermeasures, a practical and complete course of action available to the people in the bureau is called a strategy. Each person in each game participating in the countermeasure, i , $i \epsilon I$, has its own decision set D_i

(3) Win the matrix Each of the two sides meets the $x_1 \in D_1, x_2 \in D_2$ decision (x_1, x_2), using $u(x_1, x_2)$ to represent the effectiveness of the game in the game. Use the winning matrix $M = |\, u(i, j) \,|$ to represent the benefits of all game possibilities.

(4) Nash equalizer Both sides of the game strive to maximize their effectiveness in the game through decision-making, with the player represented separately 1 and player 2 will select the strategy. Look for satisfaction

$$\begin{cases} u_1({x_1}^*, {x_2}^*) \geq u_1(x_1, {x_2}^*) \\ u_2({x_1}^*, {x_2}^*) \geq u_2({x_1}^*, x_2) \\ x_1 \epsilon D_1 \\ x_2 \epsilon D_2 \end{cases} \qquad \ldots\ldots\ldots\ldots\ldots..(17)$$

The strategy combination of is the Nash equilibrium of this problem x ∗ = (x1 ∗ , x2 ∗).

5.2.2 Model solution

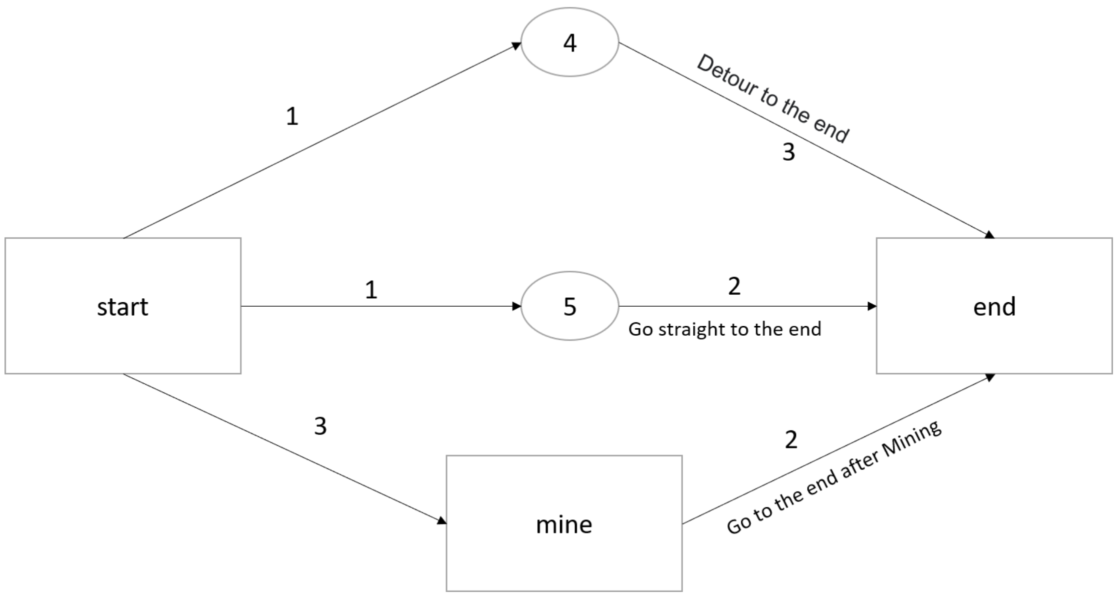


Figure 11 Level 5: Decision path selection

.

In order to arrive at the strategy, set of the people in this Council, we should first consider the situation of only one player. In the game preset in this article, the goal of the people in the game is to reach the end within the specified time and earn as much money as possible. The general optimal strategy is to go to the mine to dig, the bureau people in the path to the mine and the choice of mining date to play a game. However, the prerequisite for the fifth level is special, because the mining income is low, and the weather conditions have been given, after the dynamic planning model established above and solved, it is found that the final remaining amount of the mining strategy is 9325 yuan.

In contrast, the final remaining amount from the starting point to the end point is $9535. Under such preconditions, the best strategy for people in the bureau is no longer to mine, but to return directly to the end point by the nearest path. Then we introduce the game situation. It has been calculated that the optimal unpack capacity of the player in this question will not exceed the maximum limit, whether it is to go directly to the end point or to mine, so the player's goal is only to focus on how to make the remaining amount at the end.

Table 5 One player's strategy

Policy Number	Strategy description	Final amount
Strategy 1	Mining first and then heading to the end	￥9325
Strategy 2	Go straight to the end without mining	￥9535
Strategy 3	Go straight to the end without mining	￥9425

Maximize, i.e. you want to take a strategy of getting to the end as quickly as possible from the starting point. However, when both players in the game adopt this strategy, their respective results are calculated to be $9070 due to the limitations of the rules of the game, which is less than $9535 for going directly to the end point alone and $9325 for going to mining alone.

Table 6 Two players have the same policy

Strategy number	Strategy description	The final amount for each player
Strategy 4	Both players mine first and then head to the end	￥8515
Strategy 5	Neither player mine directly to the end point	￥9070
Strategy 6	Neither player digs a mine to make the detour to the end	￥8850

In the end, we describe game decisions as the following game issues:

- The participants in the game (people in the game) are 2 players;
- There are three strategies for the game: go directly to the end, make a detour to the end, dig and then go to the end;
- The decision set of the 2 players in the game is the same;
- The purpose of both sides of the game is to successfully reach the end and make the remaining amount the largest;

The actions of game decisions and the results they produce are shown in the following table: We mark player 1 as $x_1 \epsilon D_1 = 1, 2, 3$, respectively, indicating a direct trip to the end, a detour to the end and a mine to the end. The same player 2 possible decisions are recorded as $x_2 \epsilon D_2 = 1, 2, 3$.

The utility function of Player 1 can be used to win the matrix

$$M_1 = |\, u_1(i,\ j)\, |_{3\times 3} = \begin{pmatrix} 9070 & 9425 & 9325 \\ 9535 & 8850 & 9325 \\ 9535 & 9425 & 8515 \end{pmatrix}$$

Similarly, the winning matrix for Player 2 is

Table 7 The final amount under the two players' game

Player 2 / Player 1	Go directly to the end	Take a detour to the end	After mining, go to the end
Go directly to the end	9070, 9070	9425, 9535	9325, 9535
Take a detour to the end	9535, 9425	8850, 8850	9325, 9425
After mining, go to the end point	9535, 9325	9425, 9325	8515, 8515

$$M_2 = |\, u_2(i,\ j)\, |_{3\times 3} = \begin{pmatrix} 9070 & 9535 & 9535 \\ 9425 & 8850 & 9425 \\ 9325 & 9325 & 8515 \end{pmatrix}$$

The Nash equilibrium strategy that is easy to solve is (2, 1) or (1, 2), i.e. Player 1 chooses to go around to the end, Player 2 chooses to go directly to the end point, or Player 1 chooses to go directly to the end point, and Player 2 chooses to make a detour to the end. If decisions between players are sequencing and mutual information is fully public, then the player who makes the decision will make the decision in accordance with the Nash equilibrium once he or she learns the result of the decision of the player who made the decision first.

However, players in this question are required to make decisions at the same time and cannot communicate. Since there are 2 Nash equalizations in the above game, it is not possible to make optimal decisions. Further consideration, with probability to depict the behavior of the player, can find a hybrid strategy. Since Player 1 and Player 2 face the same conditions in this article, we only analyze Player 1's decisions. The likes of setting Player 1 to take action i are $p_i(i = 1\ ,\ 2\ ,\ 3\ )$, and the likely to set Player 2 to take action i is $q_i(i = 1\ ,\ 2\ ,\ 3\ )$,

and then the mixed policy set of Player 1 is

$$D_1 = p = (p_1,\ p_2,\ p_3) \mid 0 \le p_i \le 1\ ,\ \sum_{i=1}^{3} p_i = 1 \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(18)$$

Define the utility expectations of Player 1 under the hybrid strategy

$$U_1(p,\ q) = pM_1q^T = \sum_{i=1}^{3}\sum_{j=1}^{3} p_i m_{ij} q_j \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(19)$$

The decision-making problem of maximizing the expected utility is

$$\max \quad U_1(p\ ,\ q), p \in S_1 \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(20)$$

Solve with Lingo

$$p_1 = 0.609, \qquad p_2 = 0.301, \qquad p_3 = 0.090$$

In the case of multiple games repeated, if the player selects strategy 1 with a 60.9% probability, strategy 2 with a 30.1% probability, and strategy 3 with a 9% probability, the player's desired utility is the highest. If you only face one game, the player can only make one decision. The best strategy for maximizing utility is to get the fifth level:

Table 8 Best strategy for the fifth level

Initial purchase	81kg of water and 66kg of food
Decision path	Follow the shortest path to the end, move on day 1, stop on day 2, move on day 3, move on day 4, and reach end on day 5

5.3 The sixth pass Model is Established and solved

5.3.1 Model Building

First, we make the following assumptions about the player's behavior:

- Players are rational players; the goal is to maximize the remaining amount to win the game;
- Players do not communicate before making decisions, cannot produce collusion, cooperation;
- All players make decisions at the same time;
- Regardless of malicious competition between players, they deliberately follow the same path as other players to consume each other's resource's purpose.

The Malcolm decision-making process is used for analysis. Markov's decision-making process is a circular process in which an intelligent body takes action to change its state to be rewarded and interact with the environment. In this question, the player is regarded as intelligent, has a complete perception of the environment and conditions of the entire game, and its next action is based entirely on the current state of decision-making, reflecting the Markov nature.

We will mark Markov's decision-making process as follows: $M =< S,X,P_{s,x},R >$

(1) State collection

We use $s_k \in S_k$ to represent a collection of limited states in Phase k, which is all the information contained in the current situation for decision. Unlike the state referred to in the 3.2.2 section dynamic planning model, the state here includes not only the player's own information, but also the action plans of other players and the number of resources remaining. As in player 1, for example, the state collection is shown in Table 9:

(2) Action collection

We use $x_k \in X_k$ k stage to represent a collection of actions. The map in this question is a rectangular square, and when the player is in any of the grids of the map, it is in an initial state $S_{0,j}, j \in \{1,2\}$the next state, which can make any action down the map to the left or right to enter the next state$S'_{i,j}, i \in \{1,2,3,4\}, j \in \{1,2\}$

If other players are not considered, the current player will decide the next move according to the optimal strategy. However, if multiple players are introduced, this round of action will result in a game when the optimal decision of two or more players points to the same grid on the map, as shown in Figure 12.

Before proceeding, first analyze what choices players make in the face of game conflicts. As can be seen from the discussion of the fifth level, the hybrid strategy solves the decision probability at the maximum of the mean of return, but the strategy of any one of the games is still uncertain. Because from the trend of multi-game, all players have the same initial conditions, if all

Table 9 Status information contained in the status collection

Serial Number	Serial Information	Serial Number	Serial Information
A	Personal money	G	Player 3 money
B	My remaining resources	H	Player 3 remaining resources
C	My backpack capacity	I	Player 3 strategy of the day
D	Player 2 money	J	Weather
E	Player 2 remaining resources	K	Basic game parameters
F	Player 2 strategy of the day	L	Basic map of the game

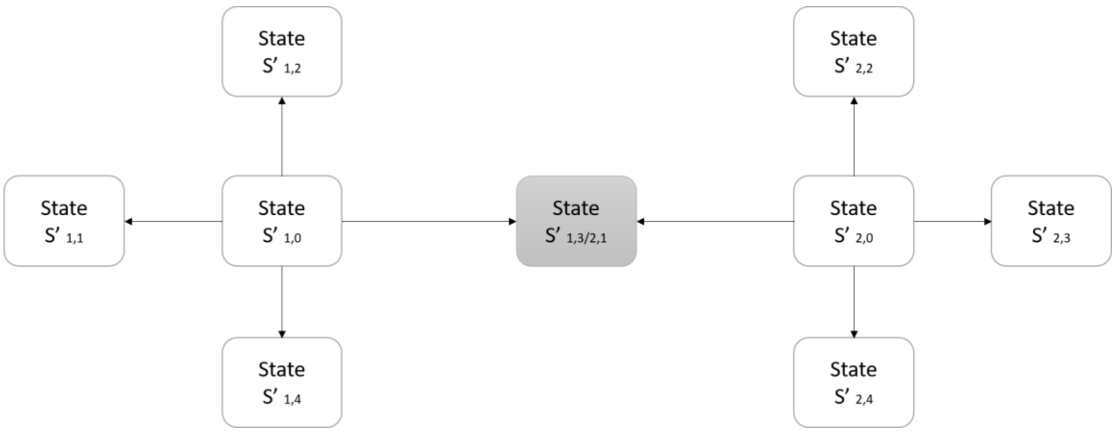


Figure 12 Level 6: Action Set and Game Diagram

Choose to make decisions with the best strategy, and after enough games, the sum of all their innings funds should gradually converge. Through simulation experiments, this inference can be verified, the construction of the fifth level of the game environment, so that player 1 and player 2 with 5.2.2 subsection of the probability to develop a strategy, repeat 5000 games, player 1's capital accounted for the proportion of the total amount of funds of two people with the change of game number of relationships as shown in Figure 13.

As you can see, as the number of games increases, the proportion of player 1's funds converges to 50% (and accordingly, the proportion of players' 2's funds converges). This shows that for a number of players who are exactly the same, no matter how they make decisions, as long as they make decisions on the same basis, the outcome of the game will eventually reach complete equilibrium.

Therefore, different properties need to be introduced for different players. When faced with a game, some players, regardless of the risk of consuming several times the resources, decided to move on, known as "risk appetite", and some players choose to be cautious, preferring to bypass rather than encounter conflicts that result in additional resource consumption, known as "risk aversion."" Different players have different levels of risk acceptance, which makes their decision-making based on different differences, the discussion of the outcome of the game will be more meaningful.

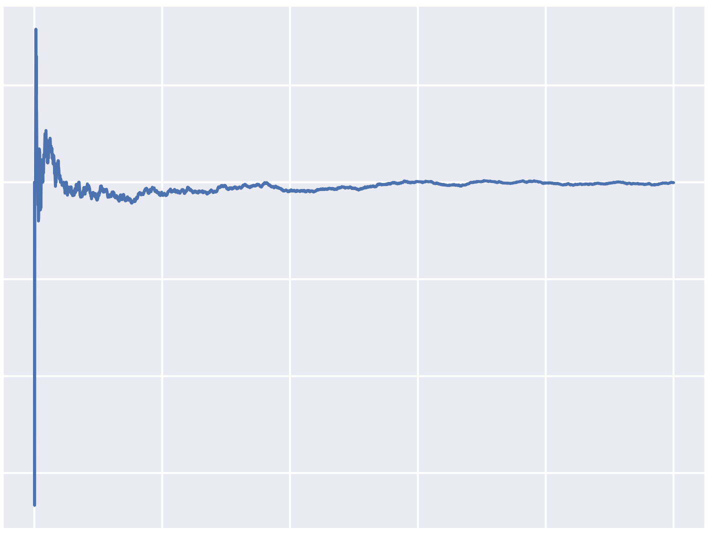


Figure 13 Repeat experiments on the fifth-level game strategy

(3) The probability of state transfer

In order to depict the player's process of considering the current state and making a decision, i.e. the probability of making an action in the state of sk, the probability of risk acceptance is introduced as the probability of state transfer

$T(S, a,S') \sim P_r(s' \mid s, a)$ ……………………..(21)

When multiple players' optimal paths conflict, if a player's optimal path is also the only viable path, he will have no choice. In addition, the probability of state transition varies for players with different risk appetites, as shown in Table 10.

Table 10 Differences in the probability of state transfer for different risk appetites

Risk appetite	Impact on the probability of state transition
Risk appetite	The player moves to the conflict point with a probability of 66.6% − 100%
Risk neutral	Player moves to the point of conflict with a probability of 33.3% to 66.6%.
Risk aversion	players move to conflict points with a 0% probability of 33.3%.

(4) Return

When a state transition occurs, the action pays off, and the benefits of a one-step action can be portrayed as $R(S, a) = E[R_{t+1} \mid s, a]$ ………………(22)

The target function of the procedure is the sum and maximum of all states, considering that the game time is 30 days, so the objective function of this question model is expressed as: $max\ U(s_0,\ s_1,\ ..., s_{30})\ =\ \sum_{t=0}^{30} R(s_t,\ a_t)$ ……………. (23)

5.3.2 Model Solution

Build an emulator using Monte Carlo simulation to simulate the Markov decision-making process of players with different levels of risk acceptance, and then explore the impact of risk acceptance on the player's decision-making.

In each simulation, the initial conditions are exactly the same for all players except risk acceptance (state transfer probability), and the risk acceptance of each player is randomly specified by the program. At each stage, the player considers whether the other player's possible course of action conflicts with their optimal sub-strategy from the current position to the end, and if the conflict is based on the level of risk acceptance, the decision to go to the point of conflict or make a detour. The solution method of the optimal sub-strategy in the simulation process is the same as that of the dynamic planning solution in question one. At the end of each simulation, record each player's decision path and the final remaining funds. If the player fails to reach the end point due to insufficient resources or funds (the game fails), the remaining funds are 0.

Analysis of the simulation results reveals that the proportion of players with the highest risk acceptance among the three players in the game is about 40% (the highest percentage), most of whom are risk arsonists and risk arsonists. However, simulation results also show that too many conflicts from excessive risk acceptance can consume too many resources and cause the game to fail. To explore the appropriate level of risk acceptance, all failed game players can be counted for risk acceptance, a histogram can be drawn and a nuclear density estimate can be made, as shown in Figure 14.

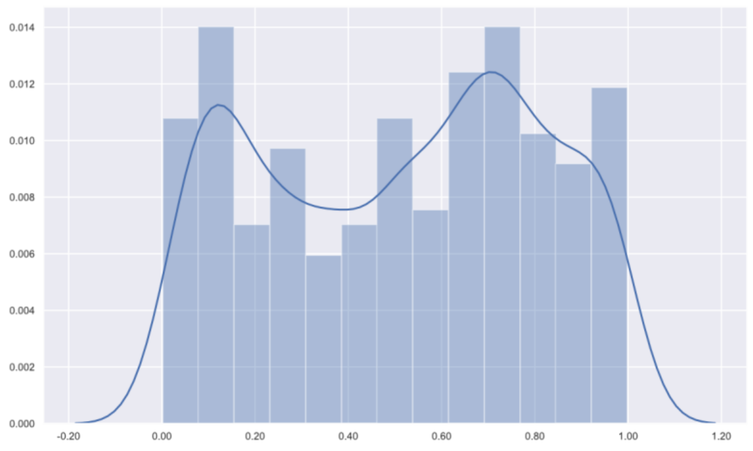


Figure 14 An estimate of the degree of risk acceptance for all failed gamers

As you can see, players with a risk acceptance of 20% to 60% are less likely to fail the game. Combining risk acceptance with the final remaining funds, maintaining a risk acceptance of around 60% is a good strategy.

Chapter 6 Model evaluation and promotion

6.1 Question One

For question one, after simplifying the original map into a directional graph model, a complete dynamic planning model, including state variables, decision variables, state transfer functions, profit and loss functions, etc., is established by analyzing the decision-making scheme and constraints in the rules of the game. In the implementation of the algorithm, combined with multiple searches, memory search, pruning and other optimization means, significantly improve the computational speed.

Based on the model and algorithm, the global optimal strategy of the first and second levels can be solved directly. The model is adaptable and stable, and under the condition of only one player, the model can be extended to any kind of map situation where the weather conditions are known.

6.2 Question 2

In the context of environmental uncertainty, the model establishes the player's decision path by analyzing the accumulation of uncertainties, by adjusting the Dijkstra algorithm, creating a simplified network with key locations as the core. At the same time, the relationship between the player's earnings and individual behaviors is established, thus establishing several basic decision-making bases, in the specific environment, only need to carry out cost analysis and environmental analysis, you can get the player's general best strategy.

The third level and the fourth level of the conditions are similar, because the key node before the player encountered the environment is inconsistent, in the key node players need to make decisions according to specific circumstances, while calculating the player's returns in different circumstances, considering the uncertainty of the environment, you can analyze the best strategy, the model can be extended to other issues.

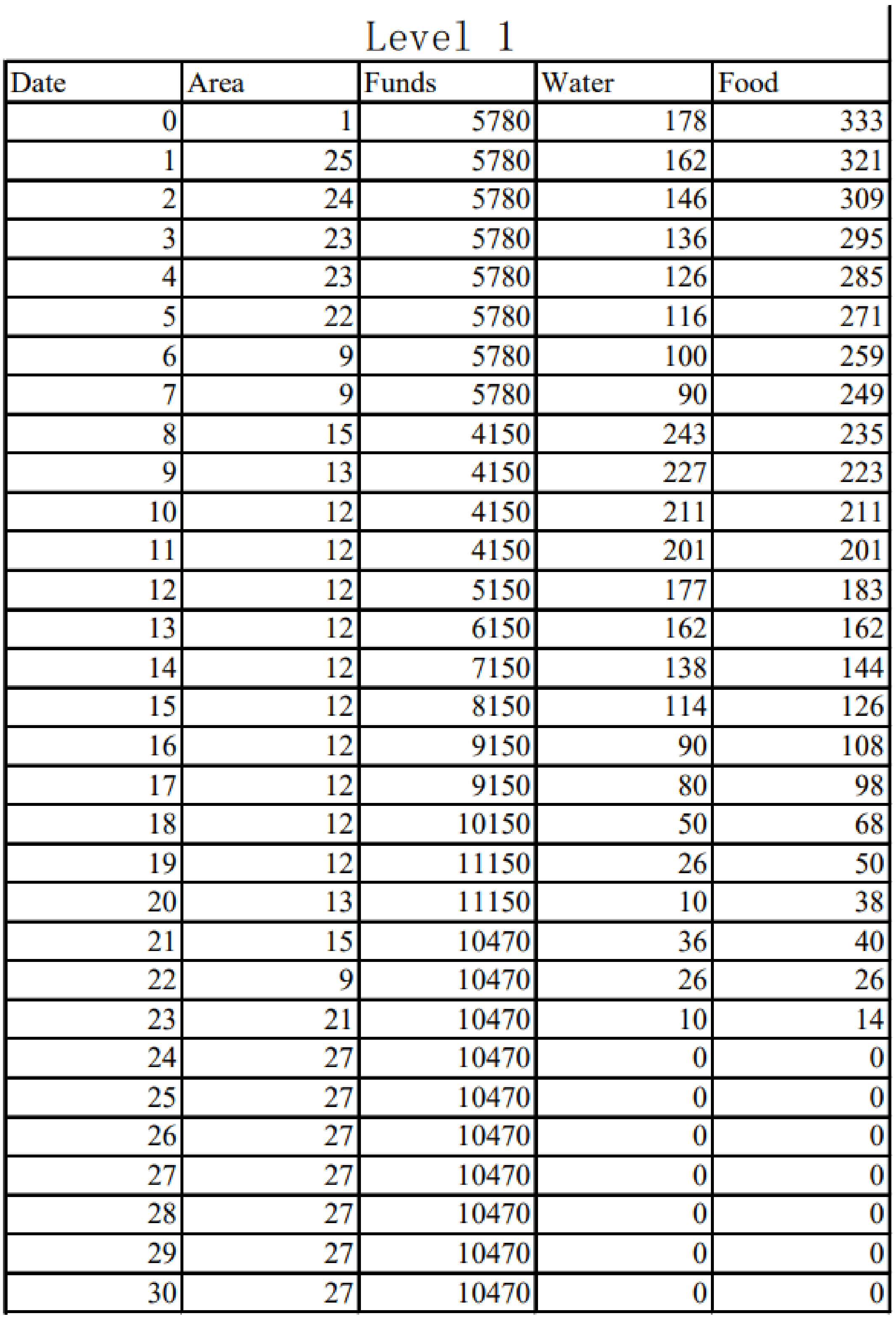
6.3 Question 3

For question three, based on the idea of static game and dynamic game, the basic model of player's behavior strategy is established based on the conclusions of the first two questions, under the circumstances of multi-player participation and the decision-making results will affect each other.

For the static full information game at level five, it is assumed that there is no exchange of information between players. The model abstracts out three decision-making choices, and calculating the probability of the strategy from the perspective of hybrid strategy can guarantee the maximum amount mean from the game of large number of disks, but the guiding significance of the decision-making of a game is limited.

For the sixth level, the introduction of risk appetite to the player's choice of tendency in the game to depict, which adapts to the actual gamer is not a completely rational person's actual situation. In the solution of the model, the model of problem one is extended, the result maximization problem of Markov's decision-making process is solved by dynamic planning, and the decision-making situation in the rules of the game is better restored. However, the model does not take into account the use of rules between players to cooperate, retaliation and other operations, in the portrayal of human factors can still be improved.

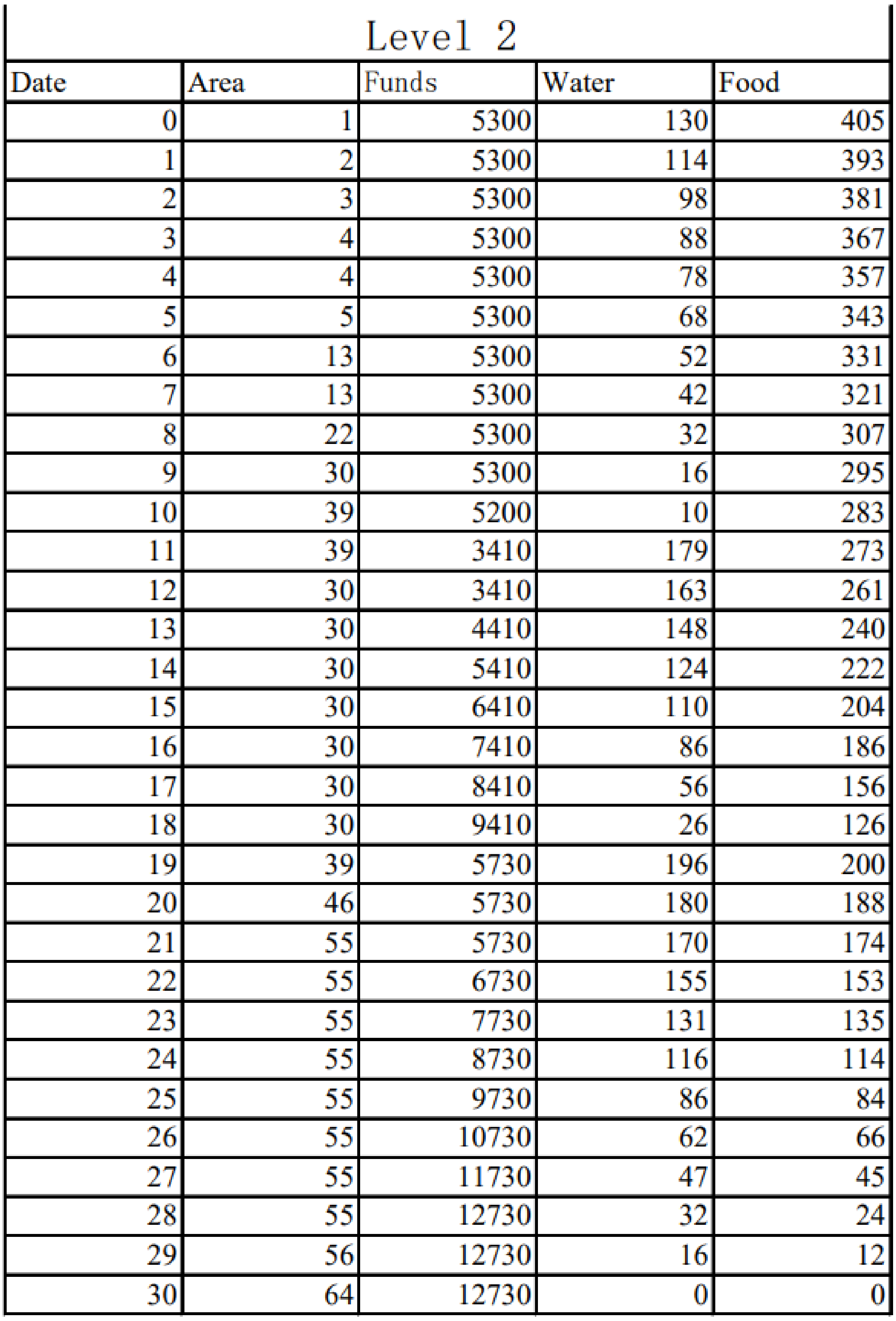
Appendix A Results of the Level 1



Level 1

Date	Area	Funds	Water	Food
0	1	5780	178	333
1	25	5780	162	321
2	24	5780	146	309
3	23	5780	136	295
4	23	5780	126	285
5	22	5780	116	271
6	9	5780	100	259
7	9	5780	90	249
8	15	4150	243	235
9	13	4150	227	223
10	12	4150	211	211
11	12	4150	201	201
12	12	5150	177	183
13	12	6150	162	162
14	12	7150	138	144
15	12	8150	114	126
16	12	9150	90	108
17	12	9150	80	98
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	27	10470	0	0
26	27	10470	0	0
27	27	10470	0	0
28	27	10470	0	0
29	27	10470	0	0
30	27	10470	0	0

Appendix B Results of Level 2



Level 2

Date	Area	Funds	Water	Food
0	1	5300	130	405
1	2	5300	114	393
2	3	5300	98	381
3	4	5300	88	367
4	4	5300	78	357
5	5	5300	68	343
6	13	5300	52	331
7	13	5300	42	321
8	22	5300	32	307
9	30	5300	16	295
10	39	5200	10	283
11	39	3410	179	273
12	30	3410	163	261
13	30	4410	148	240
14	30	5410	124	222
15	30	6410	110	204
16	30	7410	86	186
17	30	8410	56	156
18	30	9410	26	126
19	39	5730	196	200
20	46	5730	180	188
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

Appendix C Shortest Path Source Code

```
import pandas as pd
import json
import copy

#get all the intersection points of the path,
select the point closest to the target location
def getKeyPoint(routes1, routes2, pt1, pt2):
    all_inters = getAllInters(routes1,routes2)
    min_distance = 99999999
    key_pt = []
    for inter in all_inters:
        routes1 = getAllRoutes(inter,pt1)
        routes2 = getAllRoutes(inter,pt2)
        distance = len(routes1[0]) +
len(routes2[0])
        if distance < min_distance:
            key_pt = inter
            min_distance = distance
    return key_pt

#get the possible intersection of the two
shortest paths
def getAllInters(routes1,routes2):
    all_inters = []
    for route1 in routes1:
        for route2 in routes2:
            for i in range(1,len(route1)):
                if route1[i] in route2:
                    all_inters.append(route1[i])
    return all_inters

# solve all the ways according to the directed
chain
def getAllRoutes(from_id, to_id):
    print(from_id, to_id)
    front_neighbors =
getAllFrontNeighbor(from_id, to_id)
    print(front_neighbors)
    routes = []
    reversed_routes = [[to_id]]
    temp_reversed_routes = []
    i = 0
    while True:
        get_end = False
        temp_reversed_routes = []
        for route in reversed_routes:
            for front_pt in
front_neighbors[route[-1]-1]:

temp_reversed_routes.append(route+[front_pt])
                if front_pt == from_id:
                    get_end = True
        reversed_routes =
copy.deepcopy(temp_reversed_routes)
        if get_end == True:
            break
        i = i +1
    for item in reversed_routes:
        item.reverse()
        routes.append(item)
    # print(routes)
    return routes

# Get all one-way network
def getAllFrontNeighbor(from_id,to_id):
    cur_pts = [from_id] # the point outside the
current round
    searched_pts = [from_id] # locations that
have been retrieved
    around_graph = [] # graphs stored by distance
    cur_distance = 1 # Current distance
    front_neighbors = [[] for i in
range(problem.shape[0])] #there may be multiple
adjacent edges in front

    while True:
        # First calculate all the surrounding
points
        all_neighbor_pts = []
        for cur_pt in cur_pts:
            for pt_id in neighbors[cur_pt-1]:
                if pt_id not in searched_pts:
                    if pt_id not in
all_neighbor_pts:

all_neighbor_pts.append(pt_id)
                    front_neighbors[pt_id-
1].append(cur_pt)
        # After calculating the area of the round
        around_graph.append(all_neighbor_pts)
        cur_pts = all_neighbor_pts
        searched_pts = searched_pts +
all_neighbor_pts

        if to_id in searched_pts:
            break

    return front_neighbors

if __name__ == "__main__":
    problem_id = 3
    problem = pd.read_csv("data/problem" +
str(problem_id) + "_graph.csv")
    neighbors = []

    for i in range(problem.shape[0]):

neighbors.append(json.loads(problem["neighbor"][i
]))

    from_id, to_id1 = 1, 13 # Starting position
and target position
    routes1 = getAllRoutes(from_id,to_id1) # Get
all the paths
    print("all path from 1-13: ")
    print(routes1)

    from_id, to_id2 = 1, 9 # Starting position
and target position
    routes2 = getAllRoutes(from_id,to_id2) # Get
all the paths
```

```
    print("all path from 1-19: ")
    print(routes2)

    key_pt =
getKeyPoint(routes1,routes2,to_id1,to_id2)
    print("The starting point to the end point
and the potential decision-making position of the
mine: ")
    print(key_pt)
```

Appendix D Dynamic Programming Source Code

```
from random import random,choice,shuffle

class Point:
    def __init__(self,index):
        self.name=index
        self.neighbours=[]
        self.type=0
        self.players=0
        # 0 1 2 3 4:  village, mine, starting
point and ending point

    def addNeighbour(self,pt):
        self.neighbours.append(pt)

    def setType(self,t):
        self.type=t

class Solution:
    def __init__(self,step,prev,money,pt,key):
        self.step=step
        self.prev=prev
        self.next=[]
        self.money=money
        self.pt_index=pt
        self.key=key
        self.last_supply=key
        self.last_supply_pt=0
        self.last_cash=money

class Decision:
    def
__init__(self,start,end,weather,getMineral=False)
:

self.start,self.end,self.weather=start,end,weathe
r
        self.water=self.food=self.money=0 #
Consumption is positive, earning is negative.
Water or food units are boxes
        if start.name==end.name and
```

```
start.type!=4:
            if getMineral:
                self.getMineral(weather)
            else:

self.water=WATER_CONSUMPTION[weather]

self.food=FOOD_CONSUMPTION[weather]
        if start.name!=end.name:

self.water=2*WATER_CONSUMPTION[weather]
            self.food=2*FOOD_CONSUMPTION[weather]
            if MOVE_PLAN[start.name-1][end.name-
1]>1:

self.water=self.water*MOVE_PLAN[start.name-
1][end.name-1]

self.food=self.food*MOVE_PLAN[start.name-
1][end.name-1]


    def getMineral(self,weather):
        self.water=3*WATER_CONSUMPTION[weather]
        self.food=3*FOOD_CONSUMPTION[weather]
        if POINTS[self.start.name-1].players>1:
            if MINING>0:
                self.money=-PROFIT/MINING
            else:
                self.money=-PROFIT
        else:
            self.money=-PROFIT

WEATHER=[]
DAY_NUM=0
MAX_BURDEN=0
INIT_MONRY=0
PROFIT=0
WATER_WEIGHT=0
WATER_PRICE=0
FOOD_WEIGHT=0
```

```
FOOD_PRICE=0
WATER_CONSUMPTION={}
FOOD_CONSUMPTION={}
POINTS=[]
POINT_NUM=0
DESTINATION=0
MOVE_PLAN=[]
MINING=0
WATER_ADD={}
FOOD_ADD={}
MONEY_REDUCE={}

def loadPoints(file_name):
    with open(file_name, 'r') as f:
        for line in f.readlines():
            line=line.split(',')
            p=Point(int(line[0]))
            POINTS[int(line[0])-1]=p
            if line[2].replace('\n','')!='':
                p.setType(int(line[2]))
    with open(file_name, 'r') as f:
        for line in f.readlines():
            line=line.split(',')
            if line[1]!='':
                for pt in line[1].split(' '):
                    POINTS[int(line[0])-
1].addNeighbour(POINTS[int(pt)-1])

def loadEnvir(problem_no):
    global WEATHER
    global DAY_NUM
    global MAX_BURDEN
    global INIT_MONRY
    global PROFIT
    global WATER_WEIGHT
    global WATER_PRICE
    global FOOD_WEIGHT
    global FOOD_PRICE
    global WATER_CONSUMPTION
    global FOOD_CONSUMPTION
    global POINT_NUM
    global DESTINATION
    global MOVE_PLAN
    if problem_no==1 or problem_no==2:
        WEATHER='high temperature, high
temperature, sunny, sandstorm, sunny, high
temperature, sandstorm, sunny, high temperature,
high temperature, sandstorm, high temperature,
sunny, high temperature, high temperature, high
temperature, sandstorm, sandstorm, high
temperature, high temperature, sunny, sunny, high
temperature, sunny, sandstorm, High temperature,
sunny, sunny, high temperature, high
temperature'.split(',')
        DAY_NUM=30
        MAX_BURDEN=1200
        INIT_MONRY=10000
        PROFIT=1000
        WATER_WEIGHT=3
        WATER_PRICE=5
        FOOD_WEIGHT=2
        FOOD_PRICE=10
        WATER_CONSUMPTION={'high
temperature':5,'sunny':8,'sandstorm':10}
        FOOD_CONSUMPTION={'high
temperature':7,'sunny':6,'sandstorm':10}
        POINT_NUM=12 if problem_no==1 else 17
        DESTINATION=9 if problem_no==1 else 12
        assert(len(WEATHER)==DAY_NUM)
        for i in range(POINT_NUM):
            POINTS.append([])
            MOVE_PLAN.append([])
            for j in range(POINT_NUM):
                MOVE_PLAN[i].append(0)

loadPoints('data/problem{}_graph_simple.csv'.form
at(problem_no))
    if problem_no==6:
        DAY_NUM=30
        MAX_BURDEN=1200
        INIT_MONRY=10000
        PROFIT=1000
        WATER_WEIGHT=3
        WATER_PRICE=5
        FOOD_WEIGHT=2
        FOOD_PRICE=10
        WATER_CONSUMPTION={'sunny':3,'high
temperature':9,'sandstorm':10}
        FOOD_CONSUMPTION={'sunny':4,'high
temperature':9,'sandstorm':10}
        POINT_NUM=25
        DESTINATION=24
        for i in range(POINT_NUM):
            POINTS.append([])
            MOVE_PLAN.append([])
            for j in range(POINT_NUM):
                MOVE_PLAN[i].append(0)
        loadPoints('data/problem6_graph.csv')

def getDecision(point,day):
    decision_list=[]
    if WEATHER[day]!='sandstorm':
        for pt in point.neighbours:

decision_list.append(Decision(point,pt,WEATHER[da
y]))

decision_list.append(Decision(point,point,WEATHER
[day]))
    if point.type==2:

decision_list.append(Decision(point,point,WEATHER
[day],getMineral=True))
    return decision_list

def getKey(water,food):
    return str(water).zfill(4)+str(food).zfill(4)

def revertKey(key):
    return int(key[0:4]),int(key[4:8])

def
dp_main(init_water,init_food,start_day=0,init_pt=
0,init_money=None,all_path=False):
    global WATER_ADD
    global FOOD_ADD
    global MONEY_REDUCE
    solution=[]
    for i in range(DAY_NUM+1):
        solution.append([])
        for j in range(POINT_NUM):
            solution[i].append({})
    cur_key=getKey(init_water,init_food)
    if init_money==None:
        init_m=INIT_MONRY-init_water*WATER_PRICE-
init_food*FOOD_PRICE
    else:
        init_m=init_money

solution[start_day][init_pt][cur_key]=Solution(st
art_day,None,init_m,init_pt,cur_key)
    for step in range(start_day,DAY_NUM):
        real_date=step
        pt_list=list(range(POINT_NUM))
        shuffle(pt_list)
        for pt in pt_list:
            records=solution[step][pt]
            if len(records)==0:
                continue

decisions=getDecision(POINTS[pt],real_date)
            shuffle(decisions)
            for d in decisions:

_water,_food,_money=d.water,d.food,d.money
                for key in list(records.keys()):
                    cur_solution=records[key]
                    water,food=revertKey(key)
                    new_water=water-_water
                    new_food=food-_food
                    new_money=cur_solution.money-
_money

last_cash=cur_solution.last_cash
```

```

last_supply=cur_solution.last_supply

last_supply_pt=cur_solution.last_supply_pt
                    if d.end.type==4 or
d.end.type==1:

last_water,last_food=revertKey(last_supply)
                        last_cash=new_money
                        water_buy,food_buy=0,0
                        if new_water<0:
                            water_buy=-new_water
                        if new_food<0:
                            food_buy=-new_food
                        if water_buy>0 or
food_buy>0:
                            if
last_supply==cur_key:
                                continue
                            canBuy=False
                            if
POINTS[last_supply_pt].players<2:
                                cost_k=2
                            if
POINTS[last_supply_pt].players>1:
                                cost_k=4

cost=water_buy*WATER_PRICE*cost_k+food_buy*FOOD_P
RICE*cost_k
                            if
(last_water+water_buy)*WATER_WEIGHT+(last_food+fo
od_buy)*FOOD_WEIGHT<=MAX_BURDEN and
cost<=last_cash:
                                canBuy=True
                            if canBuy:

new_money=new_money-cost

new_water=new_water+water_buy

new_food=new_food+food_buy
```

```
WATER_ADD[last_supply]=water_buy

FOOD_ADD[last_supply]=food_buy

MONEY_REDUCE[last_supply]=cost

last_cash=new_money

last_supply=getKey(new_water,new_food)

last_supply_pt=d.end.name-1
                            else:
                                continue
                        else:

last_supply=getKey(new_water,new_food)

last_supply_pt=d.end.name-1

new_key=getKey(new_water,new_food)

new_solution=Solution(step+1,cur_solution,new_mon
ey,d.end.name-1,new_key)

new_solution.last_cash=last_cash

new_solution.last_supply=last_supply

new_solution.last_supply_pt=last_supply_pt

cur_solution.next.append(new_solution)
                    if new_key in
solution[step+1][d.end.name-1]:
                        if
solution[step+1][d.end.name-
1][new_key].money>new_solution.money:
                            continue
                    solution[step+1][d.end.name-
1][new_key]=new_solution
    final=solution[DAY_NUM][DESTINATION]
    final_pts=[]
    final_pt=None
    max_finals=[]
    max_final=0
    for key in final:
        water,food=revertKey(key)
        if water<0 or food<0:
            continue

max_finals.append(water*WATER_PRICE*0.5+food*FOOD
_PRICE*0.5+final[key].money)
        final_pts.append(final[key])
    for index,value in enumerate(max_finals):
        if value>=max_final:
            max_final=value
            final_pt=final_pts[index]
    with open('output.csv','a') as f:

f.write(str([init_water,init_food,max_final]).rep
lace('[','').replace(']','')+'\n')
    if not all_path:
        return final_pt,max_final
    else:
        return
final_pt,max_final,final_pts,max_finals

def dp_all(problem_no):
    # Multiple search + dynamic programming to
solve the first and second levels
    loadEnvir(problem_no)
    global_max=0
    final_pt=None
    max_ij=[0,0]
    for p in
range(0,int(MAX_BURDEN/WATER_WEIGHT)):
        for q in
range(0,int(MAX_BURDEN/FOOD_WEIGHT)):
            if
p*WATER_WEIGHT+q*FOOD_WEIGHT>MAX_BURDEN:
                continue
            final,max_final=dp_main(p,q)
            if max_final>global_max:
                global_max=max_final
                final_pt=final
                max_ij=[p,q]
    while final_pt!=None:

print(final_pt.step,final_pt.pt_index,final_pt.mo
ney,final_pt.key)
        final_pt=final_pt.prev
    print(max_ij,global_max)

def dp_game(risk):
    #Multiple static games Solve the sixth level
    water=[200,200,200]
    food=[300,300,300]
    player_pt=[0,0,0]
    money=[6000,6000,6000]
    path=[[],[],[]]
    death=[False,False,False]
    global MINING
    global WATER_ADD
    global FOOD_ADD
    global MONEY_REDUCE
    global WEATHER
    for step in range(0,30):
        next_state=[None,None,None]
        MINING=0
        WATER_ADD,FOOD_ADD,MONEY_REDUCE={},{},{}
        for p in range(0,3):
            if death[p]:
                continue

final_pt,max_final=dp_main(water[p],food[p],step,
player_pt[p],money[p])
            while final_pt!=None:

print(final_pt.step,final_pt.pt_index,final_pt.mo
ney,final_pt.key)
                if final_pt.step==step+1:
                    next_state[p]=final_pt
                    break
                final_pt=final_pt.prev
            if next_state[p]==None:
                death[p]=True
                money[p]=0
                continue
            if next_state[p].money>money[p]:
                MINING=MINING+1
        for i in range(POINT_NUM):
            POINTS[i].players=0
            for j in range(POINT_NUM):
                MOVE_PLAN[i][j]=0
        for p in range(0,3):
            if death[p]:
                continue
            next_pt_index=next_state[p].pt_index

POINTS[next_pt_index].players=POINTS[next_pt_inde
x].players+1

MOVE_PLAN[player_pt[p]][next_pt_index]=MOVE_PLAN[
player_pt[p]][next_pt_index]+1
        for p in range(0,3):
            if death[p]:
                continue

final_pt,max_final,final_pts,max_finals=dp_main(w
ater[p],food[p],step,player_pt[p],money[p],all_pa
th=True)
            best_plan,good_plan=None,None
            while final_pt!=None:
                if final_pt.step==step+1:
                    best_plan=final_pt
                    break
                final_pt=final_pt.prev
            if best_plan==None:
                death[p]=True
                money[p]=0
                continue
            conflict=False
            for other in range(0,3):
                if p==other:
                    continue
                if death[other]:
                    continue
                if
best_plan.pt_index==next_state[other].pt_index:
                    conflict=True
            if  conflict:
                if random()<risk[p]:

_water1,_food1=revertKey(best_plan.key)

_water2,_food2=revertKey(best_plan.prev.key)
                    water[p]=water[p]+_water1-
_water2
                    food[p]=food[p]+_food1-_food2
                    if best_plan.pt_index in
WATER_ADD:

water[p]=water[p]+WATER_ADD[best_plan.pt_index]
                    if best_plan.pt_index in
FOOD_ADD:

food[p]=food[p]+FOOD_ADD[best_plan.pt_index]

money[p]=money[p]+best_plan.money-
best_plan.prev.money
                    if best_plan.pt_index in
MONEY_REDUCE:
                        money[p]=money[p]-
MONEY_REDUCE[best_plan.pt_index]

player_pt[p]=best_plan.pt_index

path[p].append(best_plan.pt_index+1)
                else:
                    good_max=0
                    for index,plan in
enumerate(final_pts):
                        while plan!=None:
                            if plan.step==step+1:
                                if
```

```
plan.pt_index!=best_plan.pt_index:
                                              if
max_finals[index]>good_max:

good_max=max_finals[index]

good_plan=plan
                                         break
                                    plan=plan.prev
                        if good_plan==None:
                            good_plan=best_plan

_water1,_food1=revertKey(good_plan.key)

_water2,_food2=revertKey(good_plan.prev.key)
                        water[p]=water[p]+_water1-
_water2
                        food[p]=food[p]+_food1-_food2
                        if good_plan.pt_index in
WATER_ADD:

water[p]=water[p]+WATER_ADD[good_plan.pt_index]
                        if good_plan.pt_index in
FOOD_ADD:

food[p]=food[p]+FOOD_ADD[good_plan.pt_index]

money[p]=money[p]+good_plan.money-
good_plan.prev.money
                        if good_plan.pt_index in
MONEY_REDUCE:
                            money[p]=money[p]-
MONEY_REDUCE[good_plan.pt_index]

player_pt[p]=good_plan.pt_index

path[p].append(good_plan.pt_index+1)
              else:

_water1,_food1=revertKey(best_plan.key)
```

```
_water2,_food2=revertKey(best_plan.prev.key)
                water[p]=water[p]+_water1-_water2
                food[p]=food[p]+_food1-_food2
                if best_plan.pt_index in
WATER_ADD:

water[p]=water[p]+WATER_ADD[best_plan.pt_index]
                if best_plan.pt_index in
FOOD_ADD:

food[p]=food[p]+FOOD_ADD[best_plan.pt_index]

money[p]=money[p]+best_plan.money-
best_plan.prev.money
                if best_plan.pt_index in
MONEY_REDUCE:
                    money[p]=money[p]-
MONEY_REDUCE[best_plan.pt_index]
                player_pt[p]=best_plan.pt_index

path[p].append(best_plan.pt_index+1)
        print(path,money)
    with open('game.csv','a') as f:

f.write(str([risk[0],risk[1],risk[2],money[0],mon
ey[1],money[2]]).replace('[','').replace(']','')+
'\n')
    with open('path.csv','a') as f:
        f.write(str(path)+'\n')
    with open('weather.csv','a') as f:

f.write(str(WEATHER).replace('[','').replace(']',
'')+'\n')

def dp_game_all():
    loadEnvir(6)
    global WEATHER
    global DAY_NUM
    for i in range(1000):
        WEATHER=[]
        for i in range(DAY_NUM):
            rd=random()
            if rd<0.1:
                WEATHER.append('sandstorm')
            elif rd<0.55:
                WEATHER.append('sunny')
            else:
                WEATHER.append('high 
temperature')
        risk=[random(),random(),random()]
        print(risk)
        dp_game(risk)


if __name__ == "__main__":
    dp_all(2) # solve the first and the second 
level
    dp_game_all() # solve the sixth level
```

Appendix E Solving Mixed Strategy Lingo Source Code

```
model:
sets:
k/1..3/:p;
n/1..3/:q;
pay(k,n):M;
endsets
data:
M=9070 9425 9325
    9535 8850 9325
    9535 9425 8515;
enddata
max=money;
@for(n(j):
    money <@sum(k(i):p(i)*m(i,j));
);
@sum(k:p)=1;
End
```