2015 International Mathematical Modeling Challenge

Submitted Solution

Team Control Number

2015004

The attached document is a solution submitted in response to the 2015 International Mathematical Modeling Challenge (IM^2C) problem **'Move Scheduling'** and is provided as a reference point for participants in forthcoming IM^2C events.

Please take great care to ensure that no part of this solution is copied or plagiarised in the preparation of your solution for this year's International Mathematical Modeling Challenge.

2015 IM^{2}C Problem

Movie Scheduling

Team 2015004

Summary

Arranging a movie's shooting schedule can mean a lot of hard work. Different actors and actresses have their own schedules, various resources need to be booked, and special restrictions of some scenes must be considered. In order to offer a satisfactory solution, we have developed a model capable of finding possible schedules and selecting several better schedules based on our clients' preferences. We have also provided solutions that help our clients adjust their plan if an accident happens, as well as helping them identify the most important constraints that would most greatly affect their schedule if varied.

In the first part of our model, we search the possible shooting schedules according to the information that our clients provided, which may include the available time periods of each actor, specific sites and special resources. We sort out the various available dates by scenes, and using a 0-1 matrix to represent the availability of each scene on each of possible shooting dates. We then use the backtracking method, which is commonly used in solving constraint satisfaction problems and combinatorial search, to find arrangements that satisfy each scene's availability dates. Moreover, in order to further shorten the running time of our program, we pre-arranged the order of scenes, so that the scenes with the strictest restrictions would be considered first, enabling the program using the backtracking method to find more results more quickly.

In the second part of our model, we consider other special restrictions requested (for example, the required order of some scenes) and the general preference of our clients. We filter the solutions drawn from the previous part (which could be up to 10000 solutions), by first deleting the deficient solutions based on special restrictions, and then arraying the remaining solutions from the best to the worst. Our model can calculate the total shooting schedule length, the frequency of location changes, and required number of studios in a solution. And along with the average cost to rent a studio, the intended frequency of flexible day, the total budget, and traffic budget of our clients, we can assess the general appropriateness[1] and some characteristics of a schedule, based on which we array the solutions and provide 11 top schedules to our clients. The solutions include 5 overall best solutions, 2 solutions with least number of location changes, 2 solutions with least number of studios[2], and 2 solutions with least number of shooting days.

Extending our model, we provide ways to adjust an original schedule to various accidents. When our clients provide us with information of the original schedule they selected, accidents occurred and new restrictions, we can return a new schedule with feasible adjustments. These adjustments are achieved by running the program with the new data while leaving intact the scenes that have already happened or that are not expected to change. In this way, the computational complexity would be greatly reduced and the new schedule would involve as few changes as possible.

Lastly, we can also evaluate the most important constraints for our clients by calculating the differences in average characteristic value of top solutions before and after a constraint is changed. Having access to such information, our clients would be able to pay more attention to those important constraints and prevent major delays of their original shooting schedule.

[1]We use the characteristic value to evaluate the appropriateness of a solution. For further explanation, please see 2.2, where we explain in detail how we calculate the characteristic value of a solution.

[2] Different settings cannot be constructed or shooting simultaneously at the same studio, so more than one studio may be required.

Content

1. Problem interpretation

1.1 Problem restatement

As one of the major types of popular entertainment, motion picture has become a fast-growing industry. The constant demand for new excellent films makes effective filmmaking an important job. Therefore, it is necessary for every producer to develop a good filming schedule within certain constraints, such as the availability dates of stars and specific resources. We hope to use mathematical models to come up with this optimal schedule.

1.2 Assumptions and Justifications

1) The filming schedule has a limited time span whose starting point and ending point are both reachable.
2) The filming schedule must conform to the following restrictions, which are inflexible and cannot be violated:

i. The availability dates of some stars.
ii. The availability dates of specific sites.
iii. The availability dates for specific resources.
iv. The time required to construct and film on a list of sets.
v. Some scenes cannot be shot until after certain computer generated content is defined and other physical items are constructed.
vi. Some scenes cannot be shot until other scenes are finished. For example, if a set is finally destroyed, all other scenes related to this set are supposed to be shot before it happens.
vii. Extra time is needed for redoing some shots if they turn out to be inadequate after editing and review.
viii. Extra time is needed for making up some delay or changes.
ix. During the filming process, the director might decide to add some new scenes, which will influence the schedule.

3) The schedule breaks down time into days for further allocation.

- Filming schedules always choose “Day” as their unit, which is pretty reasonable. First, it is small enough, because most scenes require one or several days to be shot. Second, it is still flexible, because in every single day small delays can be compensated by working overtime, and night shoots can be compensated by more rests in the daytime.

4) Several scenes cannot be shot at the same time.

- Every time a scene is shot, the director must be present.

5) The construction of sets does not interfere with the filming process.

- Workers can build the sets themselves without being monitored by the director.

6) If multiple schedules are available, the studio will consider the continuity of locations.

- Since the crews do not wish to move frequently, a small number of changes in filming locations is preferred.

7) Since we can rent several filming studios, it is possible to build different settings at the same time. However, renting more filming studios will increase the cost, so we hope that these constructions do not conflict with each other.

8) Although the crew may not work every day, we should pay for them as long as the shoot is not finished. Therefore, the whole time span is not likely to be too long.

9) To deal with problems of delays and reshoots, the filming crews usually have a "flexible day" for every two weeks. On this "flexible day", they can shoot the scenes that were not completely finished before. Each day without a shooting plan can be seen as a flexible day.

1.3 The goal of modeling

According to the assumptions given above, the optimal schedule derived from our model is an arrangement that:

- satisfies all the inflexible restrictions in 2nd assumption;
- promises relatively less changes in locations;
- has relatively less conflicts in construction of different settings;
- has an appropriate time span;
- has enough flexible day for problems of delays and reshoots.

In this model, we attempt to find a schedule that can excel in all these criteria.

2 Model

2.1 Overview of the model

The whole model consists of two parts: "finding" solutions and "filtering" solutions. For each specific situation, our model first figures out all possible schedules, and then picks up the best ones among them. The model can also adjust previous schedules to deal with new accidents, such as delays in some aspects or changes in the availability of some assets.

2.2 Definition of variables

In order to build a reasonable model, we first introduced some variables:

$T_{\max}$: The maximum time length of filming schedule, which is the difference between the latest

date and the earliest date appearing in the input sheets. A movie cannot be finished if the span of shooting schedule is longer than $T_{\max}$.

$T_{\min}$: The minimum time length of filming schedule, which is the actual shooting time.

A_k : The personal schedule of actor k, which is a 0-1 array within the spread of $T_{\max}$. A '0' as the n^{th} element means that the actor cannot work on the n^{th} day. Similarly, '1' means the actor can join shooting on that day.

B_l : The availability dates of site l. We can simply regard a site as an actor, because to accomplish a shooting task, we need both the actors and the sites available at the same time. If one element is unavailable, the shooting cannot be accomplished. Thus we describe B_l in the same way as that of A_k; that is, a 0-1 array with the length of $T_{\max}$. Additionally, we define $|B|$ as the number of sites.

C_m : The availability dates of setting m. Similar to sites, a setting can be treated just like an actor. We define $|C|$ as the number of the settings.

D_n : The availability dates of special resource n, such as a helicopter or a tank, which is only available at certain time intervals. Since resources also can be treated as actors, we use A_k to record D_n to simplify the model. We define $|A|$ as the total number of actors and special resources. (The reason why we do not combine B_l and C_m with A_k is that information about sites and settings are required for other calculations in the "filtering" process.)

N_o : A set that records the time required (a.k.a. time length) to film scene o and the elements (e.g. actors, special resources, and a site or a setting) involved. N_o cannot be an empty set; it has to contain at least one actor and one site or setting. Generally, its time length cannot be changed, which already contains a little flexible time for preparation and transportation. Moreover, we consider a scene the smallest part of shooting, which means that a scene cannot be divided. When all scenes are finished, the shooting task is done. We define $|N_o|$ as the time length of this scene, which is an element of the set.

Q_i : A sequence showing a specific arrangement of all N_o s. Q_i is a time plan for all scenes.

t_{N_o} : The number of possible ways to arrange scene o, regardless of arrangements of all other

scenes.

x_i : The actual shooting time length of Q_i .

A_k^* : An "Invisible actor" that has the same character as a real actor. We can use this idea to fulfill the special requirements, such as a specific scene that can be shot in several particular time intervals. This variable depends on requirements of the clients.

E_p : A special restriction p that Q_i must conform to. For example, the restriction that N_1 must be shot before N_2 are accomplished. This variable depends on requirements of the clients.

P : A value between 0 and 1 that measures the ability of a schedule to deal with delays and reshoots. We define that in every $\frac{1}{P}$ days, one flexible day is used to compensate delays and reshoots. Small delays or quick reshoots, which need only a few hours, can be adjusted on the very day it occurs, so that the whole schedule will not be affected. If delays or reshoots cost a longer time, we can resort remaining scenes into a new schedule. So that only when delays and reshoots need one day or several days, the flexible day in the $\frac{1}{P}$ days is needed. P is an average value, which satisfies $0 \le P < 1$. This variable depends on requirements of the clients.

K_i : The number of changes in sites or settings (a.k.a. location changes) that occur in an arrangement Q_i .

K_0 : The least possible number of location changes.

L_i : The number of required filming studios.

y_i : The extra number of Q_i 's location changes. Since we want to avoid unnecessary location changes, the smaller y_i is, the better Q_i is.

z_i : The extra number of filming studios that are required. We say that the more filming studios are used, the more money is spent, and the less efficient the schedule is.

M_{budget} : Total budget of the movie, which is given by our client.

$M_{totaltraffic}$: Total traffic budget of the movie, which is also decided by our client.

$M_{trafficpertime_i}$: The average cost of a location change, which is the money spent on travelling and

transportation.

M_{studio} : The average cost of one studio, including the construction cost and art design cost. It is decided by client.

η : The budget per day. This variable builds a connection between time and money, so that we can find the equivalent of a certain amount of money in some measure of time. However, it is just an estimation variable, time is always invaluable.

θ_i : Penalty coefficient of y_i (illustrated below).

α : Penalty coefficient of z_i (illustrated below).

2.3 Relations between variables

According to the definitions of these variables, we can come up with some basic relationships:

$$T_{\min} = \sum |N_o| \qquad (2.3\text{-}1)$$

$$T_{\min} \le x_i \le T_{\max} \qquad (2.3\text{-}2)$$

$$K_0 = |B| + |C| - 1 \qquad (2.3\text{-}3)$$

$$y_i = K_i - K_0 \qquad (2.3\text{-}4)$$

$$\text{when } L_i \neq 0,\ z_i = L_i - 1;\ \text{when } L_i = 0,\ z_i = 0 \qquad (2.3\text{-}5)$$

$$M_{trafficpertime_i} = \frac{M_{totaltraffic}}{K_i} \qquad (2.3\text{-}6)$$

$$\eta = \frac{M_{budget}}{T_{\min}} \qquad (2.3\text{-}7)$$

We believe that a large number of changes in locations are “bad” for a shooting schedule; too many filming studios are “bad”, too. In order to describe the how bad they are, we can transform their financial cost into time of equal value. So we have

$$\theta_i = \frac{M_{trafficperday_i}}{\eta} \qquad (2.3\text{-}8)$$

$$\alpha = \frac{M_{studio}}{\eta} \qquad (2.3\text{-}9)$$

Moreover, when the constraints are too loose, most of possible solutions will be far from optimal. To prevent this condition, we add another limit:

$$\left\lfloor \frac{T_{\max}}{2P+1} \right\rfloor \le T_{\min} \qquad (2.3\text{-}10)$$

This means that $T_{\max}$ cannot be too long. $T_{\max}$ should be longer than $T_{\min}$ by a period of time no more than $2PT_{\min}$, or there will be too much idle time. Since too much idle time causes the increase of possible solutions, if the case holds false to the formula, we know that there are too many Q_i s.

2.4 Algorithm of finding all possible schedules

Finding possible schedules is a process of trial and error, which is meant to include both successes and failures. However, in order to reduce the mass of calculation, we hope to obtain all solutions with the least number of failures. To achieve this, we can determine the relatively inflexible times first, and leave the flexible ones for permutations later.

Since t_{N_o}, the number of possible ways to arrange scene o (regardless of all other scenes), is an indication of N_o 's flexibility in time, we can arrange all N_o s by ascending order of t_{N_o}

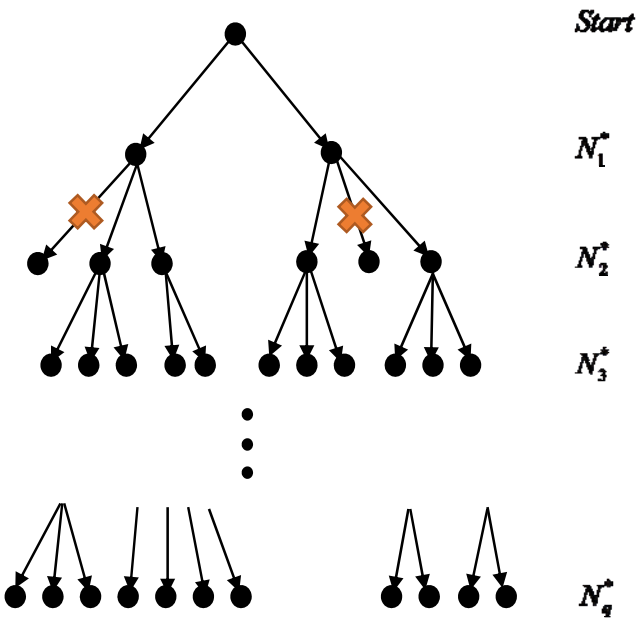
$$\left\{N_1^*, N_2^*, N_3^*, ..., N_{q-1}^*, N_q^*\right\}$$

This arrangement enables us to settle these scenes in an efficient order.

Then we use backtracking algorithm to find all of Q_i.

In fact, the finding process is like finding all $N_1^* N_q^* - path$ s in a directed graph:

First, we choose one N_o^*. Then we pick one of the available choices of N_{o+1}^*. If we cannot find a N_{o+1}^*, which means that the chosen N_o^* is unfeasible, we will go back to the last order and find another N_o^*. We will continual to repeat this procedure until we get a complete $N_1^* N_q^* - path$. Every



time we obtain a path, we record it and go back to the last order.

Even though the crotches of tree are numerous and unknown, we have promised the least number of crotches by arranging N_o by t_{N_o}. After reducing a large amount of computations, the newly derived N_o^* is suitable for finding Q_i.

2.5 Algorithm of finding the best schedule

After all possible schedules are found, we need to evaluate each of them to find the top solutions. First, we have to follow the restrictions in E. If Q_i does not accord with E_p, it should be eliminated from our consideration.

Second, we need to assess how good Q_i is. In an optimal situation, we can assume a numerical relationship among several variables:

$$x_i - T_{\min} - \theta_i y_i - \alpha z_i \approx T_{\min} P \qquad (2.5\text{-}1)$$

x_i is the whole time span of Q_i, while $T_{\min}$ is the real shooting time, and θ_i and α are the time spent on transportations and constructions. Therefore, the left side of the equation stands for the actual flexible time ready to deal with delays and reshoots, which is expected to be $PT_{\min}$

Based on formula (11), we introduce a new index $S(Q_i)$:

$$S(Q_i)=\ln\left(\frac{\left|T_{\min} P-(x_i-T_{\min}-\theta_i y_i-\alpha z_i)\right|}{T_{\min} P}\right) \qquad (2.5\text{-}2)$$

The function $S(Q_i)$, a characteristic value, indicates the superiority of each Q_i. The less $S(Q_i)$ is, the more orderly Q_i is, and the better the schedule is.

In the best solution whose free time is just appropriate, $T_{\min} P-(x_i-T_{\min}-\theta_i y_i-\alpha z_i)=0$, and $S(Q_i) \to -\infty$.

If free time is too little, and flexible days are just enough to cover the penalty time (time spent on transportations and constructions) but not enough to compensate for delays and reshoots, we will have $(x_i - T_{\min} - \theta_i y_i - \alpha z_i) = 0$, and $S(Q_i) = 0$.

If free time is too much, and the time for delays and reshoots are twice as much as needed, we will have $(x_i - T_{\min} - \theta_i y_i - \alpha z_i) = 2T_{\min} P$, the characteristic value $S(Q_i)$ also equals to 0. Even

though the two cases above are different, they are both considered as mediocre choices, and their $S(Q_i)$ values are the same.

2.6 Adjustment for accidents

If the clients encounter an accident that cannot be solved by existing flexible days (for example, significant delays in one aspect or the availability of some asset changes) during the filming process, we can provide a model for them to rearrange the future plan. The rearrangement can be achieved through a similar program, so they could quickly obtain the new schedule after changing some basic information and running the program.

This adjustment is based on information in several aspects:

1) The change in constraints (if any). For example, if the delay is caused by change in availability of an actor, a site or a resource, the new available time must be known.

2) The present achievements. Since some tasks are already finished, the schedule before the present day cannot change anymore.

3) The future dates that are hard to change. Some appointments with actors and specific resources (such as a helicopter) cannot be cancelled or changed, so the related scenes have to stay at the same date.

After these information are entered, the program will add new constraints $\left\{A_1^*, A_2^*, A_3^*, ..., A_r^*\right\}$ and the unchangeable scenes $\left\{N_1^{'}, N_2^{'}, N_3^{'}, ..., N_r^{'}\right\}$ (unchangeable scenes will be considered as "Invisible actors", and the detailed procedure is discussed in 2.2) and give solution based on the new circumstances. Therefore, if these information are known, the program can provide an optimal future schedule for the clients when an accident happens.

3 Programming Approach

3.1 Overview of the program

Based on our model, we developed a computer program that is capable of generating possible schedules as well as determining the priority of each schedules according to their characteristic values and sorting them by the priority in descending order.

3.2 Introduction of the program

3.2.1 Inputs

Our program contains a function that can read an Excel file so that it is very convenient for clients to input data. The Excel has 6 sheets which contain the information of actors, places, sets, scenes and two kinds of restrictions.
The input function named 'Read' works by transferring information in Excel into four variables (Data form: double) named Ifm_Actors, Ifm_Places, Ifm_Sets, Ifm_Scenes and Cmd_Time.

3.2.2 Algorithm of finding

3.2.2.1 Preliminary filtering
First of all, the program will check whether a possible solution exists. If there's obviously no possible solution[3], it will display an error and terminates the program.
3.2.2.2 Estimation and tips
Use the formula (10) to determine whether the restrictions are too loose so that the quantity of possible schedules might be too large. If the case holds false to the formula, the program would display a warning.
3.2.2.3 Searching for possible schedules
The program generates 0-1 matrixes Ifm_Actors, Ifm_Places, Ifm_Sets and Ifm_Scenes with 1 representing available days and 0 representing occupied days. In this way we can figure out the intersection of available time by multiplying the correspondent row.
Then, by rearranging the 0-1 matrix in order of the method that was mentioned in 2.4, the computational complicity is significantly reduced.
Finally, we use the backtracking method to search for all possible schedules within the 0-1 matrix. Solutions are saved in variable All_Solutions (data form: cell).
During the searching, if the number of solutions is too large, the program simply stops searching when 1000 solutions are recorded.

3.2.3 Algorithm of filtering

After generating all solutions, our program will follow time restrictions to delete those solutions that don't fit in. Then, the program determines the changing times of places and sets, and calculates the characteristic values with function (12) to judge whether a solution is practical enough. The solutions are rearranged according to their characteristic values.

[3] For example, if the schedules of two actors who cooperate in a specific scene do not have intersection, the scene obviously cannot be shot.

3.2.4 Outputs

Finally, the output function of the program creates an Excel file and writes 11 possible schedules with the top priority in four different criteria. They include 5 overall best schedules, 2 schedules with least number of location changes, 2 schedules with least number of studios, and 2 schedules with least number of shooting days. In each schedule, scenes are arranged chronologically.

4. Case Analysis

4.1 Overview of the case

For most movies, the detailed information about the preparing process are not open to public, which makes it hard to find a realistic case to test our model. However, we gathered some basic facts about film shooting from internet and libraries, and used them to create a mostly reliable case to test our model.
This movie tells the stories of a group of physicists. It contains 18 roles, 9 sites, 11 settings, and 29 scenes. The studio wants the whole filming to be finished within 60 days. They also give some constraints, which are all displayed as below.

4.1.1 Roles and the available times of their actors

The following chart displays all the roles included in the film and the available dates of their actors:

Role	Available dates of actor
Rydberg	13-14
Thomson	26-28
Rutherford	1-5, 24-25, 29-30
Chadwick	4-5, 29-30
Planck	11-12, 16-17, 23, 29-30
Bohr	1-3,7,9,26-28, 34-35, 51-54
Landau	21
Fermi	41-50
Oppenheimer	22, 41-47, 51-52
Feynman	44-47
Schrödinger	18-20, 34-35
Compton	24-25, 34-35
De Broglie	34-35, 40-60
Einstein	6, 8, 16, 17, 21, 34-40

Heisenberg	7-9, 23, 31-36, 53-60
Dirac	31-35, 51-52
Pauli	10, 34-35
Born	10, 19-20, 34-35

4.1.2 Scenes

The following chart displays all the scenes in the film, the locations of and roles in them, and the time needed to shoot them. There are two types of locations, sites and settings, which will be specifically discussed in following paragraphs.

Scene	Location	Roles	Time(day)
Solvay Conference	Brussels	Einstein, Bohr, Planck, Dirac, Born, Pauli, Schrödinger, Compton, De Broglie, Heisenberg	2
Experiment 1	Lab 1	Rydberg, Bohr	2
Experiment 2	The University of Manchester	Rutherford, Bohr	3
Experiment 3	The University of Cambridge	Thomson, Chadwick	3
Story 1	The University of Manchester	Rutherford, Chadwick	2
Story 2	The University of Cambridge	Rutherford, Einstein	2
Story 3	Humboldt University of Berlin	Planck, Einstein	2
Story 4	Auditorium 1	Landau	1
Story 5	The University of Chicago	Fermi, Oppenheimer, Feynman	4
Story 6	The University of Cambridge	Dirac, Heisenberg	2
Story 7	Humboldt University of Berlin	Schrödinger	1
Story 8	House 1	Einstein, Heisenberg	1
Story 9	House 2	Planck, Heisenberg	1
Story 10	Auditorium 2	Einstein	1
Story 11	Meeting room 1	Einstein	1
Experiment 4	University of Copenhagen	Dirac, Bohr, Oppenheimer	2
Experiment 5	Lab 2	Compton, Rutherford	2
Story 12	Auditorium 2	Heisenberg, Bohr	1
Story 13	University of Copenhagen	Heisenberg, Bohr	2
Story 14	University of Göttingen	Pauli, Born	1
Experiment 6	Lab 3	Planck	2
Story 15	Meeting room 2	Heisenberg, Bohr	1
Story 16	Princeton University	Einstein	2
Story 17	House 1	Einstein	2
Story 18	Humboldt University of Berlin	Schrödinger, Born	2
Story 19	House 3	Oppenheimer	1
Story 20	Lab 4	Oppenheimer, Fermi	2
Story 21	National Library	Fermi	3
Story 22	University of Copenhagen	Heisenberg	2

4.1.3 Available time of sites

For some specific reasons, a few sites cannot be shot all the time. The following chart displays the available dates of each site:

Place	Available date
University of Copenhagen	50-60
Humboldt University of Berlin	15-20
Brussels	1-60
The University of Manchester	1-7
The University of Cambridge	25-33
The University of Chicago	44-60
University of Göttingen	1-57
Princeton University	1-42
National Library	48-60

4.1.4 Preparation time of settings

The time of preparation for settings is a determinant of the number of filming studios. The following chart displays this preparation time:

Set	Preparation time(day)
House 1	4
House 2	3
House 3	6
Meeting room 1	0
Meeting room 2	0
Lab 1	0
Lab 2	0
Lab 3	0
Lab 4	0
Auditorium 1	0
Auditorium 2	0

We assume in this way because compared to filming in meeting rooms, labs and auditoriums, it is harder to film in a realistic house (For example, a high filming position cannot be reached). Therefore, we must build some “houses” in the filming studio, which requires several days to prepare.

4.1.5 Restriction of specific orders

The studio also requires some specific orders of scenes to be shot:
“Story 20” must go after “Experiment 1”.
“Story 13” must go after “Experiment 5”, and “Experiment 5” must go after “Experiment 2”.

4.1.6 Other constraints

The studio also gives the following requirements for specific reasons:
"Solvay Conference" must be shot between 30th and 35th day.
"Experiment 3" must be shot between 20th and 28th day.
"Story 2" must be shot before halfway, that is, the 30th day.

4.1.7 Other settings

In this case, the studio wants a flexible day for every two weeks. That is, P = 1/14.
The average expenditure to build a setting is approximately ￥25,000.
The total expenditure for transportation is approximately ￥420,000.
The total budget is approximately ￥13,486,000.

4.2 Case solving

4.2.1 The optimal schedule

After we entered all the constraints into Excel sheets and ran our program, we obtained a cell of 1280 29*2 matrixes that indicates 1280 different feasible schedules.
According to this solution, we derived the following overall optimal schedule.

Time(day)	Scene	Site	Setting
1-3	Experiment 2	The University of Manchester	
4-5	Story 1	The University of Manchester	
6	Story 10		Auditorium 2
7	Story 12		Auditorium 2
8	Story 11		Meeting room 1
9	Story 15		Meeting room 2
10	Story 14	University of Göttingen	
11-12	Experiment 6		Lab 3
13-14	Experiment 1		Lab 1
16- c317	Story 3	Humboldt University of Berlin	
18	Story 7	Humboldt University of Berlin	
19-20	Story 18	Humboldt University of Berlin	
21	Story 4		Auditorium 1
22	Story 19		House 3
23	Story 9		House 2
24-25	Experiment 5		Lab 2
26-28	Experiment 3	The University of Cambridge	
29-30	Story 2	The University of Cambridge	

31-32	Story 6	The University of Cambridge	
34-35	Solvay Conference	Brussels	
36	Story 8		House 1
37-38	Story 17		House 1
39-40	Story 16	Princeton University	
41-42	Story 20		Lab 4
44-47	Story 5	The University of Chicago	
48-50	Story 21	National Library	
51-52	Experiment 4	University of Copenhagen	
53-54	Story 13	University of Copenhagen	
57-58	Story 22	University of Copenhagen	

4.2.2 Schedule analysis

Comparing it with constraints mentioned above, we find that this schedule accords with all the inflexible restrictions (in 4.1.1, 4.1.3, 4.1.5 and 4.1.6).

The 3rd and 4th rows of the schedule shows a good continuity in location: all scenes related to the same place are shot successively, which promises the least number of moves.

Considering the preparation of settings, we realize that the shoot in House 3 would conflict with the construction of House 2: when House 3 is being used, we cannot prepare for House 2 for the next day at the same place. Therefore, we have to rent two filming studios and let them work at the same time when necessary. However, the time span of this schedule is good: it only lasts for 58 days, so we can end the filming process two days earlier. Meanwhile, among these 58 days, we have 5 flexible days (days 15, 33, 43, 55, 56) ready to make up unexpected delays and failures. On this score, the characteristic value of this schedule is likely to reach the minimum.

4.3 Adjustment for accidents

4.3.1 Assumed accidents

Now we assume that on the 20th day of filming, the available dates of two actors suddenly change: Oppenheimer cannot work on the 22nd day, and Einstein cannot work on 39th and 40th days anymore. However, since too many changes might cause inconvenience, the studio hopes not to change the previous schedule after 41st day. To make up this delay, they have told all the actors to get ready for reshoots from 55th to 60th day.

4.3.2 Adjustment realization

To make a rearrangement, the client can follow these steps:

1. Enter the changes in time constraints (of Oppenheimer, Einstein and other actors) into the input

file.

2. Retain the lines of unchangeable scenes (scenes on days 1-20 and days 40+), and delete all other lines in the previous "schedule" file. Save it as "schedule_delay.xlsx".

3. Run the program.

A new optimal "schedule" file will be generated based on the present condition.

4.3.3 Results

After the rearrangement, the new optimal schedule is generated as below:

Scenes	Start time	End time
Experiment 2	1	3
Story 1	4	5
Story 10	6	6
Story 15	7	7
Story 8	8	8
Story 12	9	9
Story 14	10	10
Experiment 6	11	12
Experiment 1	13	14
Story 3	16	17
Story 7	18	18
Story 18	19	20
Story 4	21	21
Story 9	23	23
Experiment 5	24	25
Experiment 3	26	28
Story 2	29	30
Story 6	32	33
Solvay	34	35
Story 16	36	37
Story 11	38	38
Story 20	41	42
Story 19	43	43
Story 5	44	47
Story 21	48	50
Experiment 4	51	52
Story 13	53	54
Story 17	55	56
Story 22	57	58

It is clear that this new schedule satisfies all the constraints mentioned above.

5 The way of finding the most important constraints

5.1 Overview of the algorithm

We have developed a way to find the most important constraints by calculating the difference of the top 5 solutions' average characteristic value. The more this average value changed with a constraint, the more important this constraint is.

5.2 Description of the algorithm

Based on our model, we have developed an algorithm to determine the most important constraints. We assume that the shooting has not begun, so that our model is not affected by any events.
The algorithm can be achieved by the following steps:

1) Compute the average value of the top 5 solutions' characteristic values $\overline{S(Q_i)_0}$.

2) Choose an element (one actor, place, or set).

3) For every element, define W_{Q_i} as the number of following changes that result in no possible schedule.

4) Randomly select one day in the element's available time and delete it.

5) If we still get possible solutions, compute the average value of the top 5 solutions' characteristic values $\overline{S(Q_i)_{1.1}}$ after the change[4].

6) Repeat step 3), step 4), and step 5) for other $v-1$ times to obtain

$$\left\{\overline{S(Q_i)_{1.1}}, \overline{S(Q_i)_{1.2}}, \overline{S(Q_i)_{1.3}}, ..., \overline{S(Q_i)_{1.v}}\right\}$$

and W_{Q_i} where v is a manually-set parameter. Increasing v will both improve the searching precision and increase the searching time.

7) Compute the total value of W_{Q_i}.

8) Compute the average value of every defined $\overline{S(Q_i)_1}$

$$\overline{S(Q_i)_1} = \frac{\left\{\overline{S(Q_i)_{1.1}}, \overline{S(Q_i)_{1.2}}, \overline{S(Q_i)_{1.3}}, ..., \overline{S(Q_i)_{1.v}}\right\}}{v} \qquad (5.2\text{-}1)$$

9) Compute the average changes of characteristic value ΔS_1.

[4] If there are less than five solutions, we still take the average value of their characteristic value and continue the calculation.

$$\Delta S_1 = \left(\overline{S(Q_i)_1} - \overline{S(Q_i)_0} \right) \qquad (5.2\text{-}2)$$

Changing other elements, we can obtain a sequence $\{\Delta S_1, \Delta S_2, \Delta S_3, ..., \Delta S_u\}$ which represents how great an influence the change has on each constraint, as well as a sequence. $\{W_1, W_2, ..., W_u\}$. Obviously, there is

$$u = |A| + |B| + |C| \qquad (5.2\text{-}3)$$

10) We then compare the $\{W_1, W_2, ..., W_u\}$, and regard the constraint with the largest W as the most important constraint, because this element is inflexible to most randomly selected scenes.

11) If there is more than one elements with largest $\{W_1, W_2, ..., W_u\}$, we can find the most important constraint by finding the maximum value among the $\{\Delta S_1, \Delta S_2, \Delta S_3, ..., \Delta S_u\}$ of these elements, since the change of this constraint causes the characteristic value to increase significantly and therefore limits the feasibility of the new schedule.

6 Reviews and Prospect

During this research, we did not find an existing algorithm that can handle the arrangement of a movie shooting schedule. The mean difficulty is that too many conditions and restrictions ought to be considered at the same time. The uncertainty of available days is also hard to solve.

In fact, it is the initial manual experiments that gave us the inspiration of a feasible algorithm and further improvement work. We used the method of backtracking algorithm, which can find solutions orderly and quickly. Although sometimes our program is not fast enough, it manages to simulate the human approach to this work. Therefore, it seems hard to come up with any improvement for the algorithm of this model.

As for the main problem of providing optimal filming schedules, we have given a complete solving method and have made a program package that works well. Using our model, we not only can deal with problems of delays and reshoots, but also can analyze the most important constraints in filmmaking. However, the time of computation is somewhat long, so there may still be some room for improvement.

Besides, we believe that other problems may be solved using our model as an original model. For example, we can possibly use it to deal with many single-thread matching problems. Applying our model to other scheduling problems may also be fruitful.

7 Reference Material

- 《Matlab 语言即学即会》 陆宁 机械工业出版社 ISBN 7-111-07958-2
- 《影视制片项目管理》 巩继程 河北教育出版社 ISBN 978-7-5434-6295-3

8 Appendix

The following program was written in Matlab.

1.Main Program

Program 1-General Circumstances

```
Mtotaltraffic=input('Traffic Budget=');
Mbudget=input('Total Budget=');
Mshed=input('Cost to rent a new studio=');
p=input('Frequency of flexible days=');
Filename=input('Input the file name (For example: Example.xlsx):','s');
P=1/p;

[Ifm_Actors,Ifm_Places,Ifm_Sets,Ifm_Scenes,Cmd_Time,Scene_Names]=Read(Filename);
Check_Warning(Ifm_Actors,Ifm_Places,Ifm_Scenes,P);
[All_Solutions,IX]=AllSolution(Ifm_Actors,Ifm_Places,Ifm_Scenes);
All2=Delete(Cmd_Time,All_Solutions);
CS=Cond_Sets(All_Solutions,Ifm_Sets,Ifm_Actors,Ifm_Places,Ifm_Scenes);
CT=Cond_Trans(All_Solutions,Ifm_Scenes);
[All_Solutions,En]=SortEntropy(All_Solutions,CT,CS,Ifm_Scenes,P,Mtotaltraffic,Mbudget,Mshed );
All_Solutions=Sort_Cond(All_Solutions,CT(2,:));
CT1=Cond_Trans(All_Solutions,Ifm_Scenes);
good_results = Write(All_Solutions,IX,Ifm_Scenes,Ifm_Sets,Filename,CT,CS,Scene_Names);
```

Program 2-Delays & Adjustments

```
Mtotaltraffic=input('Traffic Budget=');
Mbudget=input('Total Budget=');
Mshed=input('Cost to rent a new studio=');
p=input('Frequency of flexible days=');
Filename=input('Input the file name (For example: Example_delay.xlsx):','s');
Filename2=input('Input the file name (For example: schedule_delay.xlsx):','s');
sheetname=input('Overall Best Solution No.1','s');
P=1/p;

[Ifm_Actors,Ifm_Places,Ifm_Sets,Ifm_Scenes,Cmd_Time,Scene_Names]=Read_delay(Filename,Fil
ename2,sheetname);
Check_Warning(Ifm_Actors,Ifm_Places,Ifm_Scenes,p);
[All_Solutions,IX]=AllSolution(Ifm_Actors,Ifm_Places,Ifm_Scenes);
All2=Delete(Cmd_Time,All_Solutions);
CS=Cond_Sets(All_Solutions,Ifm_Sets,Ifm_Actors,Ifm_Places,Ifm_Scenes);
```

```
CT=Cond_Trans(All_Solutions,Ifm_Scenes);
[All_Solutions,En]=SortEntropy(All_Solutions,CT,CS,Ifm_Scenes,1/14,Mtotaltraffic,Mbudget,Mshe
d );
All_Solutions=Sort_Cond(All_Solutions,CT(2,:));
CT1=Cond_Trans(All_Solutions,Ifm_Scenes);
good_results = Write(All_Solutions,IX,Ifm_Scenes,Ifm_Sets,Filename,CT,CS,Scene_Names);
```

2.Functions

Input Functions

Function 1-Read

```
function [A,P,S,T,Cmd_Time,Scene_Names] = Read(filename)
  [A,A1]=get_ifm(filename,'actors');
sA=size(A);

  [P,P1]=get_ifm(filename,'places');

  [S,S1]=get_ifm(filename,'sets');

  [~,~,RAW1]=xlsread(filename,'scenes');
Traw = {};
sRAW1 = size(RAW1);
Traw = RAW1(2:sRAW1(1),2:sRAW1(2));
Scene_Names = RAW1(2:sRAW1(1),1);
T1=RAW1(2:sRAW1(1),1);
sTraw = size(Traw);
Traw=Find_Rep(Traw,A1);
Traw=Find_Rep2(Traw,P1,0);
Traw=Find_Rep2(Traw,S1,1);

Traw = cell2mat(Traw);
Traw(isnan(Traw)==1) = 0;
T=Traw;
sT=size(T);

  [In_A,In_A1]=get_ifm(filename,'restrictions2');
A=MatAdd(A,In_A);
sI=size(In_A1);
```

```
for i=1:1:sI(1)
      for j=1:1:sRAW1(1)-1
            if strcmp(In_A1{i,1},RAW1{j+1,1})
                  for k=3:1:sT(2)
                        if T(j,k)==0
                              T(j,k)=sA(1)+i;
                              break
                        end
                  end
            end
      end
end

  [~,~,Cmd_Time]=xlsread(filename,'time restrictions');
Cmd_Time=Find_Rep(Cmd_Time,T1);

Cmd_Time=cell2mat(Cmd_Time);
Cmd_Time(isnan(Cmd_Time)==1)=0;

end
```

Function 2-Read_delay

```
function [A,P,S,T,Cmd_Time,Scene_Names] = Read_delay(filename,filename2,sheetname)
  [A,A1]=get_ifm(filename,'actors');
sA=size(A);

  [P,P1]=get_ifm(filename,'places');

[S,S1]=get_ifm(filename,'sets');

  [~,~,RAW1]=xlsread(filename,'scenes');
Traw = {};
sRAW1 = size(RAW1);
Traw = RAW1(2:sRAW1(1),2:sRAW1(2));
Scene_Names = RAW1(2:sRAW1(1),1);
T1=RAW1(2:sRAW1(1),1);
sTraw = size(Traw);
Traw=Find_Rep(Traw,A1);
Traw=Find_Rep2(Traw,P1,0);
Traw=Find_Rep2(Traw,S1,1);

Traw = cell2mat(Traw);
Traw(isnan(Traw)==1) = 0;
```

```
T=Traw;
sT=size(T);

[In_A,In_A1]=get_ifm(filename,'restrictions2');
A=MatAdd(A,In_A);
sI=size(In_A1);
T=RowAdd(T);
sT=size(T);
for i=1:1:sI(1)
    for j=1:1:sRAW1(1)-1
        if strcmp(In_A1{i,1},RAW1{j+1,1})==1
            for k=3:1:sT(2)
                if T(j,k)==0
                    T(j,k)=sA(1)+i;
                    break
                end
            end
        end
    end
end
extended_A=i;

 [In_A,In_A1]=get_ifm(filename2,sheetname);
A=MatAdd(A,In_A);
sI=size(In_A1);
In_A=In_A(:,1:2);
T=RowAdd(T);
sT=size(T);
for i=1:1:sI(1)
    for j=1:1:sRAW1(1)-1
        if strcmp(In_A1{i,1},RAW1{j+1,1})==1
            for k=3:1:sT(2)
                if T(j,k)==0
                    T(j,k)=sA(1)+extended_A+i;
                    break
                end
            end
        end
    end
end

 [~,~,Cmd_Time]=xlsread(filename,'time restrictions');
Cmd_Time=Find_Rep(Cmd_Time,T1);
Cmd_Time=cell2mat(Cmd_Time);
```

```
Cmd_Time(isnan(Cmd_Time)==1)=0;
end
```

Function 3-get_ifm

```
function [ F,F1 ] = get_ifm( filename,sheetname )
  [F,F1,~]=xlsread(filename,sheetname);
sF1=size(F1);
F1=F1(2:sF1(1),:);
F(isnan(F)==1)=0;
End
```

Function 4-Find_Rep

```
function [ output ] = Find_Rep( T,A1 )
sT=size(T);
sA1=size(A1);
for i=1:1:sT(1)
      for j=1:1:sT(2)
            for k=1:1:sA1(1)
                  if strcmp(T{i,j},A1{k,1})==1
                        T{i,j} = k;
                  end
            end
      end
end
output=T;
end
```

Function 5-Find_Rep2

```
function [ output ] = Find_Rep2( T,A1,x )
sT=size(T);
sA1=size(A1);
for i=1:1:sT(1)
      for j=1:1:sT(2)
            for k=1:1:sA1(1)
                  if strcmp(T{i,j},A1{k,1})==1
                        T{i,j} = [x,k];
                  end
            end
      end
end
```

```
output=T;
end
```

Searching Functions

Function 1-AllSolution

```
function [ output,IX ] = AllSolution( A,P,T )
global Opt;
mA=max([A(:);P(:)]);
s=size(A);
t=size(T);
a=normalize(A,mA);
p=normalize(P,mA);
N = ones(t(1),mA);
check=ones(1,mA);
for i=1:1:t(1)
    if T(i,2)==0
        N(i,:) = p((T(i,3)),:);
    end
    for j=4:1:t(2)
        if T(i,j)~=0
            N(i,:) = N(i,:) .* a((T(i,j)),:);
        else
            break
        end
    end
end
for i=1:1:t(1)
    if check'*N(i,:)==0
        Str=strcat('Hi! Dear custumer!! No solution!!!Wrong line:',num2str(i));
        error(Str);
    end
end

IX=[];
TT=T;
NN=N;
[T,IX]=Rearrange(T,TT,NN);
[N,IX]=Rearrange(N,TT,NN);

Nprim = zeros(t(1),mA);
```

```
zanshi = ones(1,mA);
m1 = 1;
Opt = zeros(t(1),2);
i = 1;
j = 1;
flag=0;
output={};
    point=0;
    while i<=t(1) && point<=10000
        while j<=mA-T(i,1)+1 && point<=10000
            Nprim(i,:) = N(i,:) .* zanshi(1,:);
            flag=0;
            if panduan(i,j,Nprim,T) == 1
                [zanshi] = jilu(zanshi,i,j,T);
                if i == t(1)
                    %Opt
                    if floor(point/10)==point/10 || point<5000
                        output=[output,Opt];
                    end
                    point=point+1;
                    zanshi = quxiao(zanshi,i);
                    j = Opt(i,1)+1;
                    flag=1;
                    break
                else
                    i = i+1;
                    j = 1;
                    flag=1;
                    break
                end
                flag=0;
            else
                j = j+1;
            end
        end
        if j>mA-T(i,1)+1 && flag==0
            if i==1
                disp('end');
                break
            else
                zanshi = quxiao(zanshi,i-1);
                i = i-1;
                j = Opt(i,1)+1;
            end
```

```
            end
            flag=0;
        end
end
```

Function 2-quxiao

```
function [zanshi] = quxiao(zanshi,i)
global Opt
for m=Opt(i,1):1:Opt(i,2)
        zanshi(1,m) = 1;
end
end
```

Function 3-panduan

```
function[a] = panduan(i,jj,Nprim,T)
a = 1;
for j=jj:1:jj+T(i,1)-1
        if Nprim(i,j) == 0
            a = 0;
            break
        end
end
```

Function 4-jilu

```
function [zanshi] = jilu(zanshi,i,j,T)
        global Opt;
        m1 = j;
        m2 = j+T(i,1)-1;
        for m=m1:1:m2
            zanshi(1,m) = 0;
        end
        Opt(i,1)=m1;
        Opt(i,2)=m2;
End
```

Function 5-Check_Warning

```
function [    ] = Check_Warning( A,P,T,p )
Tmax=max([A(:);P(:)]);
Tmin=0;
```

```
sT=size(T);
for i=1:1:sT(1)
    Tmin=Tmin+T(i,1);
end
if floor((Tmax-Tmin)/Tmin)>2*p
    disp('Warning:The number of solutions might be too large');
end
end
```

Function 6-Rearrange

```
function [ Y,Z ] = Rearrange( X,T,N );
t=size(T);
f=size(X);
sizeN=size(N);
D=zeros(t(1),ceil(sizeN(2)/2));
LEFT=zeros(t(1),ceil(sizeN(2)/2));
RIGHT=zeros(t(1),ceil(sizeN(2)/2));
Q=zeros(t(1),sizeN(2)+2);
Q(:,1)=0;
Q(:,sizeN(2)+2)=0;
w=1;
m=1;
for i=1:1:sizeN(2)
    Q(:,i+1)=N(:,i);
end

for i=1:1:t(1)
    for j=2:1:(sizeN(2)+1)
        if Q(i,j)==1 && Q(i,j-1)==0
            LEFT(i,w)=j-1;
            w=w+1;
        end
        if Q(i,j)==1 && Q(i,j+1)==0
            RIGHT(i,m)=j-1;
            m=m+1;
        end
    end
    w=1;
    m=1;
end

for i=1:1:t(1)
    for j=1:1:(ceil(sizeN/2))
```

```
            if LEFT(i,j)~=0
            D(i,j)=RIGHT(i,j)-LEFT(i,j)+1;
            end;
      end
end

Days=zeros(t(1),ceil(sizeN(2)/2));
for i=1:1:t(1)
      for j=1:1:ceil(sizeN(2)/2)
            if D(i,j)>=T(i,1)
                  Days(i,j)=D(i,j)-T(i,j)+1;
            end
      end
end

Dayssum=zeros(t(1),2);
for i=1:1:t(1)
      Dayssum(i,1)=i;
      Dayssum(i,2)=sum(Days(i,:));
end
Dayssum=sortrows(Dayssum,2);

Y=zeros(t(1),f(2));
for(i=1:1:t(1))
      a=Dayssum(i,1);
      Y(i,:)=X(a,:);
end

Z=zeros(t(1),2);
for (i=1:1:t(1));
      Z(i,1)=i;
      Z(i,2)=Dayssum(i,1);
end

end
```

Sorting & Filtering Functions

Function 1-Delete

```
function [output] = Delete(Req,imput)
B=imput;
Optnum=size(B);
```

```
Reqnum=size(Req);
for i=1:1:Optnum(2)
    P=B{1,i};
    for j=1:1:Reqnum(1)
        for k=2:1:Reqnum(2)
            if Req(j,k)~=0
                x=Req(j,k-1);
                y=Req(j,k);
                if P(x,1)>P(y,1)
                    B{1,i}=[];
                end
            end
        end
    end
end
s=size(B);
output={};
for i=1:1:s(2)
    if size(B{1,i})~=0
        output=[output,B{1,i}];
    end
end
end
```

Function 2-Cond_Trans

```
function [ output ] = Cond_Trans( imput,T )
c=size(imput);
d=size(T);
m=0;
n=0;
for i=1:1:d(1)
    if T(i,2)==0 && T(i,3)>m
        m=T(i,3);
    end
end
for i=1:1:d(1)
    if T(i,2)==1 && T(i,3)>n
        n=T(i,3);
    end
end
output=zeros(2,c(2));
for i=1:1:c(2)
    P=imput{i};
```

```
    p=size(P);
    x=0;
    Q=zeros(p(1),3);
    Q(:,1)=P(:,1);
    Q(:,2)=P(:,2);
    for j=1:1:p(1)
        Q(j,3)=j;
    end
    Q=sortrows(Q,1);
    for j=2:1:p(1)
        if ~(T(Q(j,3),2)==T(Q(j-1,3),2) && T(Q(j,3),3)==T(Q(j-1,3),3))
        x=x+1;
        end
    end
    %{
for j=1:1:p(1)
        if P(j,1)~=min(P(:,1))
            for k=1:1:p(1)
                if P(k,2)==P(j,1)-1 && (~(T(j,2)==T(k,2) && T(j,3)==T(k,3)))
                    x=x+1;
                end
            end
        end
    end
%}
    output(1,i)=x;
    x=x-m-n+1;
    output(2,i)=x;

end
```

Function 3-n_sets

```
function [n] = n_sets(T,Opt,Set,t_lgth)
mCt = max(Set(:,1)); %mCt = max C time
sS=size(Set);
sOpt = size(Opt);

S=cell(1,sS(1));
for i=1:1:sOpt(1)
    if T(i,2)==1
        j=T(i,3);
        S{1,j}=[S{1,j};[Opt(i,1),Opt(i,2)]];
    end
```

```
end
Time=zeros(sS(1),2);
for i=1:1:sS(1)
    Time(i,:)=[min(S{1,i}(:))+mCt-Set(T(i,3),1),max(S{1,i}(:))+mCt];
end
t=normalize(Time,t_lgth+mCt);
if max(Add_Column(t))~=0
    n = max(Add_Column(t))-1;
end
end
function [ Sort,ArrEntropy] = SortEntropy( Cell,Trans1,Trans2,T,P,Mtotaltraffic,Mbudget,Mshed )
Optnum=size(Cell);
Sort=cell(1,Optnum(2));
Entropy1=zeros(Optnum(2),2);
for i=1:1:Optnum(2)
    Entropy1(i,1)=i;
    Y=Entropy(i,Cell{1,i},Trans1,Trans2,T,P,Mtotaltraffic,Mbudget,Mshed);
    Entropy1(i,2)=Y;
end
ArrEntropy=sortrows(Entropy1,2);
for i=1:1:Optnum(2)
    a=ArrEntropy(i,1);
    Sort{1,i}=Cell{1,a};
end
end
```

Function 4-Cond_Sets

```
function [ output ] = Cond_Sets( input,Sets,A,P,T )
t_lgth=max([A(:);P(:)]);
s=size(input);
output=zeros(1,s(2));
for i=1:1:s(2)
    Opt=input{i};
    output(i)=n_sets(T,Opt,Sets,t_lgth);
end

end
```

Function 5-Entropy

```
function [ output ] = Entropy( n,Imput,Trans1,Trans2,T,P,Mtotaltraffic,Mbudget,Mshed)
x=max(Imput(:,2))-min(Imput(:,1));
```

```
Ki=Trans1(1,n);
S1=Trans1(2,n);
S2=Trans2(1,n);
Tmin=sum(T(:,1));
theta=(Mtotaltraffic*Tmin)/(Mbudget*Ki);
alpha=(Mshed*Tmin)/Mbudget;
d=Tmin*P-x+Tmin+theta*S1+alpha*S2;
output=log(abs(d)/(Tmin*P));
end
```

Function 6-SortEntropy

```
function [ Sort,ArrEntropy] = SortEntropy( Cell,Trans1,Trans2,T,P,Mtotaltraffic,Mbudget,Mshed )
Optnum=size(Cell);
Sort=cell(1,Optnum(2));
Entropy1=zeros(Optnum(2),2);
for i=1:1:Optnum(2)
    Entropy1(i,1)=i;
    Y=Entropy(i,Cell{1,i},Trans1,Trans2,T,P,Mtotaltraffic,Mbudget,Mshed);
    Entropy1(i,2)=Y;
end
ArrEntropy=sortrows(Entropy1,2);
for i=1:1:Optnum(2)
    a=ArrEntropy(i,1);
    Sort{1,i}=Cell{1,a};
end
end
```

Function 7-Sort_Cond

```
function [ output ] = Sort_Cond( input_All,input_C )
s=size(input_C);
output=cell(1,s(2));
[C,IX]=sort(input_C);
for i=1:1:s(2)
    output{1,i}=input_All{1,IX(1,i)};
end
end
```

Output Function-Write

```
function [good_results] = Write(All,IX,T,Set,filename,CT,CS,Scene_Names)
a=11;
```

```
sT=size(T);
good_results = cell(a);
SN={};
for i=1:1:sT(1)
    SN{i,1}=Scene_Names{IX(i,2),1};
end

sAll = size(All);
Time = [];
for i=1:1:sAll(2)
    Time(1,i) =max( max(All{i}))-min(min(All{i}));
end

Result = cell(sT(1)+1,4);
Result{1,1} = 'scenes';
Result{1,2} = 'start time';
Result{1,3} = 'end time';
Result{1,4} = 'prepare time';

All2 = Sort_Cond(All,CT(2,:));%place
All3 = Sort_Cond(All,CS(1,:));%studio
All4 = Sort_Cond(All,Time(1,:));%time
All{6}=All2{1};
All{7}=All2{2};
All{8}=All3{1};
All{9}=All3{2};
All{10}=All4{1};
All{11}=All4{2};
for i=1:1:a
    Opt = All{i};
    [sort_Opt,index] = sortrows(Opt);

    sIX=size(IX);
    for j=1:1:sIX(1)
        reIX(IX(j,2),1)=j;
        reIX(IX(j,2),2)=IX(j,2);
    end

    for j=1:1:sT(1)
        Result{j+1,1}=SN{(index(j,1)),1};
        Result{j+1,2}=sort_Opt(j,1);
        Result{j+1,3}=sort_Opt(j,2);
        if T(index(j,1),2)==1
            Result{j+1,4}=Set(T(index(j,1),3),1);
```

```
            else
                Result{j+1,4}=0;
            end
        end
        good_results{i} = Result;
        if i<=5
        xlswrite('schedule.xlsx',Result,['Overall Best Solution No.' num2str(i)

]);
        end
        if i>=6 && i<=7
        xlswrite('schedule.xlsx',Result,['Least Studios Solution No.' num2str(i-

5) ]);
        end
        if i>=8 && i<=9
        xlswrite('schedule.xlsx',Result,['Least Days Solution No.' num2str(i-7)

]);
        end
        if i>=10 && i<=11
        xlswrite('schedule.xlsx',Result,['Least Traveling Solution No.' num2str

(i-9) ]);
        end
end
```

Auxiliary Functions

Function 1-RowAdd

```
function [ Mat_out ] = RowAdd( Mat )
s=size(Mat);
Mat_out=zeros(s(1),s(2)+1);
for i=1:1:s(1)
for j=1:1:s(2)
Mat_out(i,j)=Mat(i,j);
end
end
end
```

Function 2-DeBlank

```
function [ output ] = DeBlank( input )
s=size(input);
output={};
for i=1:1:s(2)
    if size(input{1,i})~=0
        output=[output,input{1,i}];
    end
end
end
```

Function 3-DeColumn

```
function [ output ] = DeColumn( input_mat,input_num )
input_mat=input_mat';
s=size(input_mat);
input_mat=DeRow(input_mat,input_num);
output=input_mat';
end
```

Function 4-DeColumn

```
function [ output ] = DeRow( input_mat,input_num )
s=size(input_mat);
output=[];
for i=1:1:s(1)
    if i~=input_num
        output=[output;input_mat(i,:)];
    end
end
end
```

Function 5-Add_Column

```
function [ output ] = Add_Column( input )
s=size(input);
output=zeros(1,s(2));
for i=1:1:s(2)
    for j=1:1:s(1)
        output(1,i)=output(1,i)+input(j,i);
    end
```

```
end
end
```

Function 6-MatAdd

```
function [ M ] = MatAdd( M1,M2 )
s1=size(M1);
s2=size(M2);
if s1(2)==s2(2)
      M=[M1;M2];
end
b=max(s1(2),s2(2));
a=s1(1)+s2(1);
M=zeros(a,b);
for i=1:1:s1(1)
   for j=1:1:s1(2)
       M(i,j)=M1(i,j);
   end
end
for i=1:1:s2(1)
   for j=1:1:s2(2)
           M(i+s1(1),j)=M2(i,j);
   end
end
end
```

Function 7-normalize

```
function [ output ] = normalize( input,mA )
      s=size(input);
      output=zeros(s(1),mA);
      for i=1:1:s(1)
            for j=1:1:s(2)/2
                  if input(i,2*j-1)~=0 && input(i,2*j)~=0
                        for k=input(i,2*j-1):1:input(i,2*j)
                              output(i,k)=1;
                        end
                  end
            end
      end
end
```

3.User Guide

Usage

This software (named 'Program') is designed for movie makers in need of a better shooting schedule. It automatically generates possible schedules according to the available days of actors and places, the preparation time of sets, the actors and places each scene involves and time restrictions.

Input Format

The users should provide the following information:
-A Excel file with 6 sheets containing the information of actors, places, sets, scenes and time restrictions. (Note: this file must be stored at the same file with 'Program')
-The name of the Excel file. (For example, 'The Final Case.xlsx')
-total budget
-traffic budget
-Cost to rent a new studio
-Frequency of flexible days

Sheet1: Actors
Column1-The name of each actor
Remaining columns-Time range of each actor. The two adjacent cells represent a time interval in which the actor is available

Sheet2: Places
Column1-The name of each place
Remaining columns-Time range of each place. The two adjacent cells represent a time interval in which the place is available

Sheet3: Sets
Column1-The name of each set
Column2-Preparation time of the set

Sheet4: Scenes
Column1- The name of each scene
Column2-Time needed for shooting the scene
Column3-The place or set the scene takes place
Remaining columns-The actors involved in the scene

Sheet5: Time restrictions
Each line includes a time series in which the scene must happen chronologically

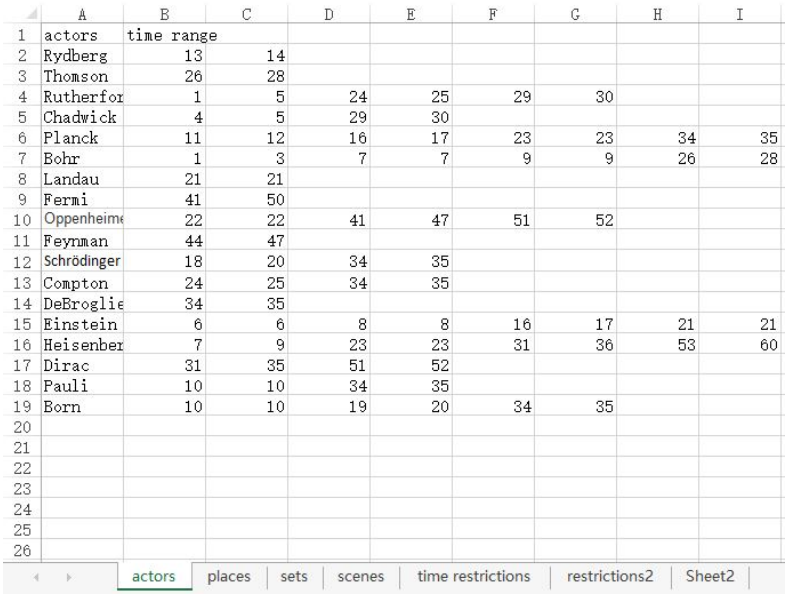
Sheet6: Restrictions2

Column1-The name of each scene

Remaining columns-Time range of each scene. The two adjacent cells represent a time interval in which the scene is available.

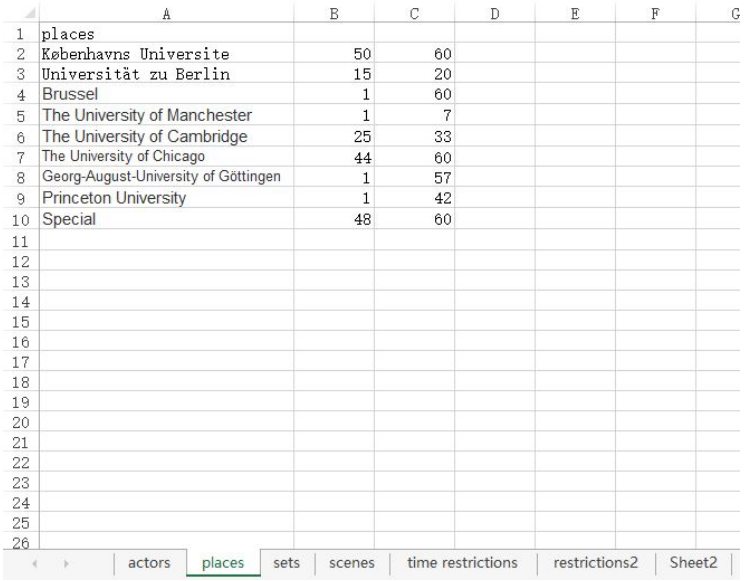
4.Examples

Input



	A	B	C	D	E	F	G	H	I
1	actors	time range							
2	Rydberg	13	14						
3	Thomson	26	28						
4	Rutherfor	1	5	24	25	29	30		
5	Chadwick	4	5	29	30				
6	Planck	11	12	16	17	23	23	34	35
7	Bohr	1	3	7	7	9	9	26	28
8	Landau	21	21						
9	Fermi	41	50						
10	Oppenheime	22	22	41	47	51	52		
11	Feynman	44	47						
12	Schrödinger	18	20	34	35				
13	Compton	24	25	34	35				
14	DeBroglie	34	35						
15	Einstein	6	6	8	8	16	17	21	21
16	Heisenber	7	9	23	23	31	36	53	60
17	Dirac	31	35	51	52				
18	Pauli	10	10	34	35				
19	Born	10	10	19	20	34	35		
20									
21									
22									
23									
24									
25									
26									

actors | places | sets | scenes | time restrictions | restrictions2 | Sheet2



	A	B	C	D	E	F	G
1	places						
2	København Universite	50	60				
3	Universität zu Berlin	15	20				
4	Brussel	1	60				
5	The University of Manchester	1	7				
6	The University of Cambridge	25	33				
7	The University of Chicago	44	60				
8	Georg-August-University of Göttingen	1	57				
9	Princeton University	1	42				
10	Special	48	60				
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							

actors | places | sets | scenes | time restrictions | restrictions2 | Sheet2

	A	B	C	D	E	F	G	H
1	sets							
2	house1	4						
3	house2	3						
4	house3	6						
5	meetingroom1	0						
6	meetingroom2	0						
7	lab1	0						
8	lab2	0						
9	lab3	0						
10	lab4	0						
11	auditorium1	0						
12	auditorium2	0						
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								

actors | places | sets | scenes | time restrictions | restrictions2 | Sheet2

	A	B	C	D	E	F	G
1	scene	time	place	people			
2	Solvay	2	Brussel	Einstein	Bohr	Planck	Dira
3	experiment1	2	lab1	Rydberg			
4	experiment2	3	The University of Manchester	Rutherfor	Bohr		
5	experiment3	3	The University of Cambridge	Thomson	Bohr		
6	story1	2	The University of Manchester	Rutherfor	Chadwick		
7	story2	2	The University of Cambridge	Rutherfor	Chadwick		
8	story3	2	Universität zu Berlin	Planck	Einstein		
9	story4	1	auditorium1	Landau	Einstein		
10	story5	4	The University of Chicago	Fermi	Oppenheim	Feynman	
11	story6	2	The University of Cambridge	Dirac	Heisenberg		
12	story7	1	Universität zu Berlin	Schrödinger			
13	story8	1	house1	Einstein	Heisenberg		
14	story9	1	house2	Planck	Heisenberg		
15	story10	1	auditorium2	Einstein			
16	story11	1	meetingroom1	Einstein			
17	experiment4	2	Københavns Universite	Dirac	Bohr	Oppenheimer	
18	experiment5	2	lab2	Compton	Rutherford		
19	story12	1	auditorium2	Heisenber	Bohr		
20	story13	2	Københavns Universite	Heisenber	Bohr		
21	story14	1	Georg-August-University of Göttingen	Pauli	Born		
22	experiment6	2	lab3	Planck			
23	story15	1	meetingroom2	Heisenber	Bohr		
24	story16	2	Princeton University	Einstein			
25	story17	2	house1	Einstein			
26	story18	2	Universität zu Berlin	Schrödinger	Born		

actors | places | sets | scenes | time restrictions | restrictions2 | Sheet2

	A	B	C	D	E	F	G
1	experiment1	story20					
2	experiment2	experiment5	story13				
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							

actors | places | sets | scenes | time restrictions | restrictions2 | Sheet2

	A	B	C	D	E	F	G	H
1	scene	After	Before					
2	Solvay	30	35					
3	experiment1	13	20					
4	experiment2	1	5					
5	experiment3	20	28					
6	story1	1	10					
7	story2	1	30					
8	story3	16	60					
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								

actors | places | sets | scenes | time restrictions | restrictions2 | Sheet2

Output

	A	B	C	D	E	F	G	H	I	J
1	scenes	start tim	end time	prepare time						
2	experimer	1	3	0						
3	story1	4	5	0						
4	story10	6	6	0						
5	story15	7	7	0						
6	story8	8	8	4						
7	story12	9	9	0						
8	story14	10	10	0						
9	experimer	11	12	0						
10	experimer	13	14	0						
11	story3	16	17	0						
12	story7	18	18	0						
13	story18	19	20	0						
14	story4	21	21	4						
15	story19	22	22	0						
16	story9	23	23	0						
17	experimer	24	25	0						
18	experimer	26	28	0						
19	story2	29	30	0						
20	story6	31	32	0						
21	Solvay	34	35	0						
22	story17	36	37	0						
23	story11	38	38	0						
24	story16	39	40	0						
25	story20	41	42	6						
26	story5	44	47	3						

... | Overall Best Solution No.5 | Least Studios Solution No.1 | Least Studios Solution No.2 | Lea ...