Team Control Number
1924188
Problem Chosen
D

For office use only
T1 ____________
T2 ____________
T3 ____________
T4 ____________

For office use only
F1 ____________
F2 ____________
F3 ____________
F4 ____________

2019
MCM/ICM
Summary Sheet

Summary

Evacuation plans are needed for many public large buildings. In this article, we build an evacuation model based on the actual situations of the Louvre, and propose policy and procedural recommendations for emergency management of the Louvre.

First we simplify the problem to a one - layer discrete model and then extend it to a multi-layer model. Also, we take visitors speaking different languages, groups traveling together, and disabled visitors into consideration to help them get out of the building swiftly, and we plan the path for emergency personnel to enter the building as well. In addition, compared with simulation software Pathfinder, we analyse the strengths, weaknesses and sensitivity of our model. Finally, further improvement of our future model is summarized and we can conclude the advice to the administrators of the Louvre.

As for the algorithms in the model, we mainly use: cellular automaton(CA), ant colony optimization(ACO), Dijkstra algorithm and A* algorithm. With cellular automaton, the original problem is turned into a discrete and solvable model, and to solve it, we use ant colony optimization algorithm to find the best path. However, the traditional ant colony optimization algorithm has some shortcomings: slow convergence and easy to fall into an infinite loop, which can be solved by using Dijkstra algorithm as heuristic function instead of simply using Manhattan distance. Besides, we design a penalty mechanism of pheromone in ACO to avoid congestion. Apart from the algorithms used for the evacuation model, we apply A* algorithm to the path planning for emergency personnel.

After running our model, the evacuation process, the time spent and the distribution of pheromone in ACO could be gained under different conditions: different people amount, different evacuating speed, different emergency situation, etc. Though the results we get from our model are not as perfect as the simulation software Pathfinder, they are close to the optimal solution and are good enough to apply in the Louvre. In our model, we also consider the situation that emergency personnel enter the museum at the same time, and optimally plans their path, which cannot be done by simulation software.

In conclusion, the Louvre administratorcan get evacuation advice from the model and make decisions according to the simulation process, for instance, which additional exit should be utilized and when to use it. And the model map can be automatically generated by the imageimported so it has a strong generalizationability.

Keywords: Evacuation Simulation, Algorithm, Pheromone

Contents

1 Introduction

Architecture is the treasure of human civilization which can prevent people from bad weather outside as well as save space for human living. With the development of society, large-scale venues and large-scale gatherings have become more and more, and the structure and function of buildings and the distribution of people have gradually become more complicated. Thus, there are more and more safety problems compared with open places. Besides some nature disasters like earthquake, terrorists also like to take architecture as their target, for the reason that intensive crowd in buildings makes attacks easier. Therefore, the evacuation of people in an emergency is an important part for the development of smart buildings in the future, especially for the complexity of the unpredictable surrounding environment, so that the best evacuation plan can be found in real time.

As one of the best-known museums, the Louvre receive more than 8.1 million visitors in 2017. What's more, with a complicated structure, it covers approximately 72,735 square meters. Also, in the last few years, there have been more terror attacks all over the world, especially in France. As a consequence, the Louvre needs a complete emergency evacuation plan in order to find one of the best options which can help visitors get out of the museum rapidly.

Taking the diversity of visitors – speaking a variety of languages, groups traveling together, and disabled visitors – into consideration, our team build a model that can guide most people to find the quickest route to the outside. Furthermore, our model can also help emergency personnel enter the building as quickly as possible, which makes solving the problem much faster. Following is the thorough description of our solution.

2 Assumptions and Justifications

We make some general assumptions to simplify our model. These assumptions together with corresponding justification are listed below:

- **The modeling environment is a discrete two-dimensional grid map**

  We assume that the building can be divided into a number of grids, each of which is a cell and has different state, so that our modeling environment is a discrete grid map, making analysis easier compared with a consequent map.Also, we assume that the structures of different floors are almost the same, except for the position of stairs or doors.

- **Each person occupies one cell on the grid map and assumed to be almost the same.**

  Because the grid area is about $4m^2$ due to our division, it is reasonable to put one person in a cell. Then we build the model considering that all people have the same speed, and the diversity of visitors will be discussed in section 4.3.

- **Visitors stay in the Louvre for about three hours and visit routes are random.**

  We hardly know exactly how many people are in the museum and where they are when the accident takes place, thus we use the 2017 data, 8 million visitors a year, to get the average amount of people per day and distribute these people randomly. Also, we can estimate instantaneous quantity of visitors using the assumed visit time 2hs.42mins(2).

- **People have their instincts guiding their evacuation routes**

  When visitors look for their directions to get out of the museum by themselves, it is trendy that they will go to the nearest exit from them, and they'd like to run with other people.

- **Simplify the map of The Louvre, especially the entrances.**

  Four entrances are mentioned in the problem, the pyramid entrance, the Passage Richelieu entrance, the Carrousel du Louvre entrance and the Portes Des Lions entrance. The location is showed in the left figure and through digital image processing methods, we simplify the map to the right figure with the entrances showed in green.

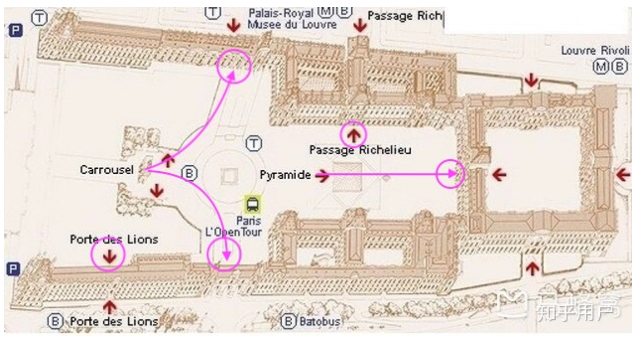


Figure 1: real map

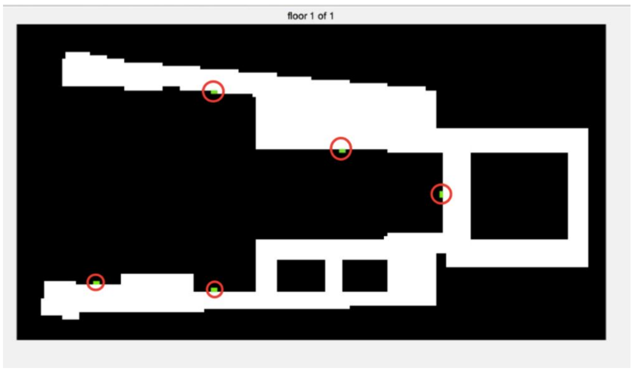


Figure 2: map simplification

3 Model

3.1 Basic Theory of Model

3.1.1 Cellular automaton(CA)

As an effective tool for the system, cellular automaton (CA) has attracted much attention because of its simple regularity of unit composition, the locality of interaction between units,the high parallelism of information processing and the complexity. We divided the whole building into a discrete grid map, which can be considered as a CA model.

In this problem, we use Moore Neighborhoods (as the Figure 3 shows) for the two-dimensional CA. Besides a discrete grid map, the model regards each person as an single cell.

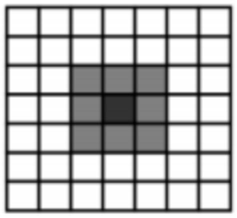


Figure 3: Moore Neighborhoods

When it comes to three dimensions, if the space is quantified, a three-dimensional vector is used to represent a unique position in the space. Since the human activity space has only 5 layers, the z coordinate has a value range of [0,5]. Then, restrict z coordinate so that it changes only at stairs. Then we can simplify a single floor to a two-dimensional evacuation, depending on the effect of the heuristic function and the pheromone.

3.1.2 Improved Ant Colony Optimization(ACO)

The early ant colony algorithm model uses the ant's walking path to represent the feasible solution of the problem to be optimized, all the paths of the entire ant group constitute the solution space of the problem to be optimized. Ants with shorter paths release more pheromone. After a few simulations, the concentration of pheromone accumulated on shorter paths gradually increases, and ants are concentrated in shorter paths. In view of the shortcomings of the common ant colony optimization, such as slow search speed and easy to fall into the "infinite loop", our model improves schemes such as target heuristic function.

The **heuristic function** in the traditional ACO algorithm is usually the reciprocal of the distance between nodes, but the distance between adjacent nodes in the CA model is constant, which has no heuristic effect and cannot be applied well. For the heuristic function, generally, the closer the person is to the exit, the larger the value is. This feature is consistent with the essence of the heuristic function. Improved heuristic function is

$$\eta_i = \frac{1}{L} \tag{1}$$

Where L is the distance to the entrance provided by Dijkstra algorithm. We can get Dijkstra distance from

$$D(v) = min(D(v), D(w) + c(w, v)) \tag{2}$$

$c(w, v)$ is the link cost of node x to y(if two nodes are not directly connected, $c(w, v) = \infty$), and $D(v)$ is the current path cost value from source to destination v.

As for the **pheromone updating**, the ant-cycle model uses global information to update the amount of pheromone on the path, which has better search performance than the other models. The updating function of ant-cycle model is

$$\tau_{ij}^k(t + \Delta t) = \begin{cases} \frac{Q}{L_k} & \text{ant } k \text{ passes route(i,j)} \\ 0 & \text{else} \end{cases} \tag{3}$$

Considering the influence of crowded people around one door, they may stay at the previous position, so our improved pheromone concentration function is

$$\tau(t + \Delta t) = (1 - \rho) \times \tau_{ij}(t) + \sum_{k=1}^{m} \tau_{ij}^k(t + \Delta t) \tag{4}$$

$$\tau_{ij}^k(t + \Delta t) = \begin{cases} \frac{1}{L_k} \times (\frac{1}{2})^{n-1} & \text{person } k \text{ passes route (i,j) in this circle} \\ 0 & \text{else} \end{cases} \tag{5}$$

Where: $\tau_{ij}^k(t + \Delta t)$ is the amount of pheromone on the path (i, j) at time (t+Δt),ρ is the coefficient of volatilization; $\tau_{ij}(t)$ is the amount of pheromone on the path (i, j) at time t; $\Delta\tau_{ij}^k$ is the pheromone increment left on the path (i, j) during the period Δt of the k; L_k is the number of cells passed by the person k in this tour; n is the the number of times person k in this tour has passed route(i,j) in this tour.

The value of ρ ranges from 0 to 1. Therefore, as the number of iterations increases, the pheromone concentration will become lower and lower on the path where people rarely pass and the path with long evacuation time, and the attraction to people will become smaller and smaller; The pheromone concentration will increase after positive feedback, and people will increasingly prefer to use shorter paths to evacuate.

Furthermore, we also change the **prohibition principle** in ACO, which is not suitable for an evacuation model. Following is our new principle guiding people. In CA, each cell is occupied by people, obstacles or is empty. A cell can accommodate at most one person, and a person cannot enter a cell that has been occupied. If more than one person competes for the same free cell at the same time, one person is randomly selected to enter, the other person stays in the original position or find another cell around him. When other cells around the person are occupied, he will stay in the present position.

In the first iteration, each person select the next cell according to the maximum initiating function value. Sort all of 8 positions and make the best choice. If the next cell is occupied, select the sub-optimal one, and so on.

In the other iterations except last one, people select the next cell according to the probability showed in the formula

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_i j(t)]^{\alpha} \times [\eta_i j(t)]^{\beta}}{\sum_{\omega \in J_i} [\tau_i \omega(t)]^{\alpha} \times [\eta_i \omega(t)]^{\beta}} & \text{if } j \in \mathrm{J}_i \\ 0 & \text{else} \end{cases} \tag{6}$$

$\tau_i j(t)$ is calculated by equations (4) and (5); $\eta_i j(t)$ is calculated by equation (3); α is the influence factor of pheromone; β is the influence factor of heuristic function; J_i is the set of empty cells near the cell ; ω is the element in the set J_i.

In the last iteration, we sort the probability given by equation(6), guiding people to choose the cell of largest possibility. Also, if the cell is occupied, select the sub-optimal one, and so on.

3.1.3 Penalty Mechanism

The traditional ant colony optimization algorithm is not well converged. The antmay move back and forth at a certain position without advancing because ofrandomness; similar situations may occur when the exitis congested.

To this end, we introduce a penalty mechanism intothe model. If the antmovesback and forth multiple times on a certain path, the pheromone on the path will volatilize at a faster rate, so that it will be selected less frequently in subsequent iterations. In this way, we can speed up the convergence of the model, and it is also possible to make the path planned by the pheromone avoid the congested area.

3.2 Model Results

3.2.1 Single-layer Result

In order to make our model closer to the actual situation of the museum, we considered the changes in the number of people in the Louvre at different times.

First we estimate the number of people in the Louvre. In the year 2017, 8.1 million(1) people visited the Louvre, 75% of whom visited during the(2). The Louvre is open about

300 days a year(except every Monday and holidays), with a rough estimate ratio of 1:2 for weekends and weekdays. Also, the museum is open for 8 hours one day and each visitor stays in the museum for about $2h40min(2.67h)^{(2)}$.Then we do some calculation about how many people are in the Louvre at some moment.

- average in a year: about 1800
- maximum in vacation: about 4050
- minimum in weekdays: about 670

Simulate separately and assume that the visitors' moving speed is about 5m/s in the emergency situation. (distance between each two grids is 2m by discretion)

When the number of people is average, the simulation process and pheromone distribution in the process are as follows:

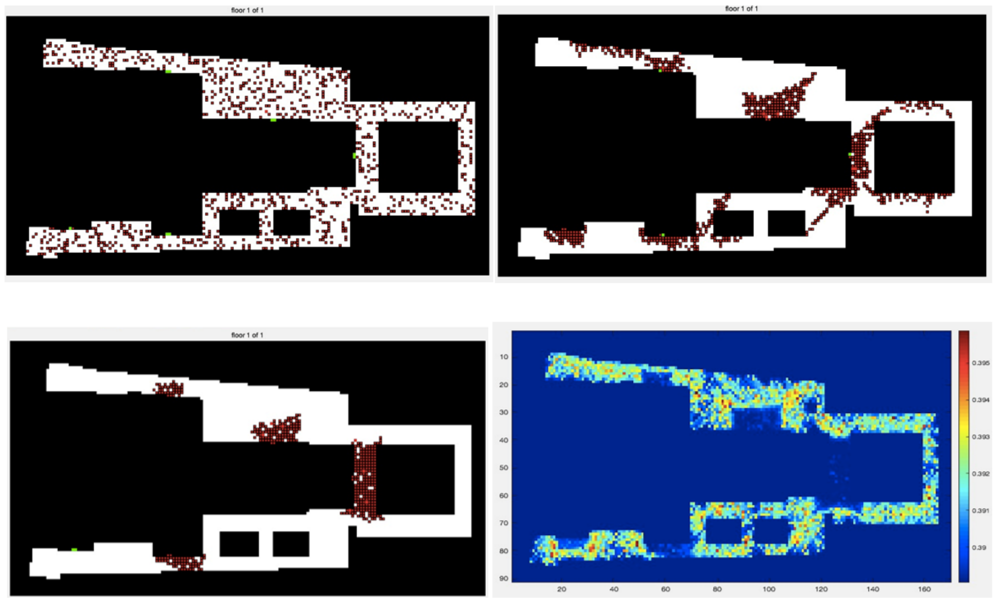


Figure 4: Average number Simulation

The model runs 168 steps to evacuate all the people. From the assumption, the total evacuation time is about 67.2s.

When the number of people is maximum, the simulation process and pheromone distribution in the process are as follows:

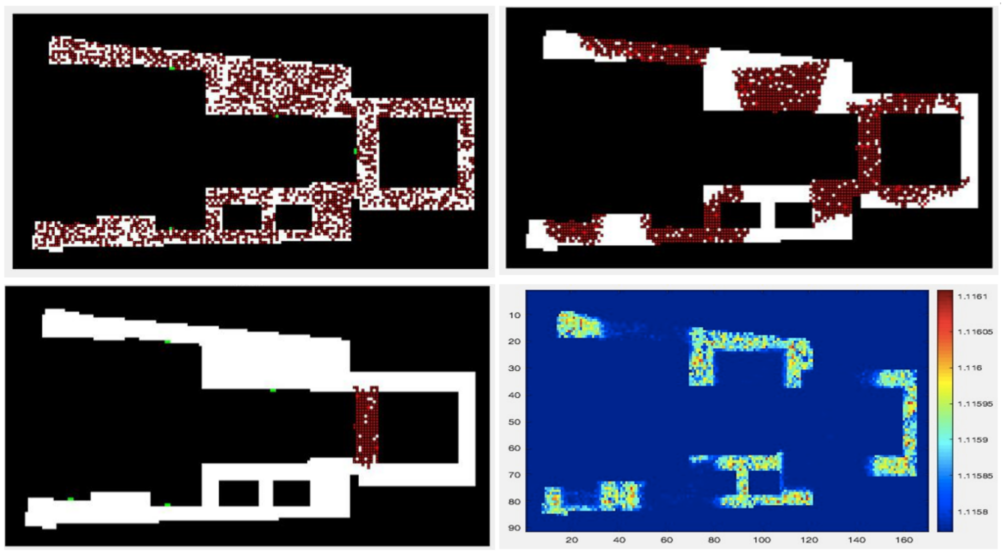


Figure 5: Max number Simulation

The model runs 380 steps to evacuate all the people. From the assumption, the total evacuation time is about 152s.

When the number of people is minimum, the simulation process and pheromone distribution in the process are showed in Figure 6.

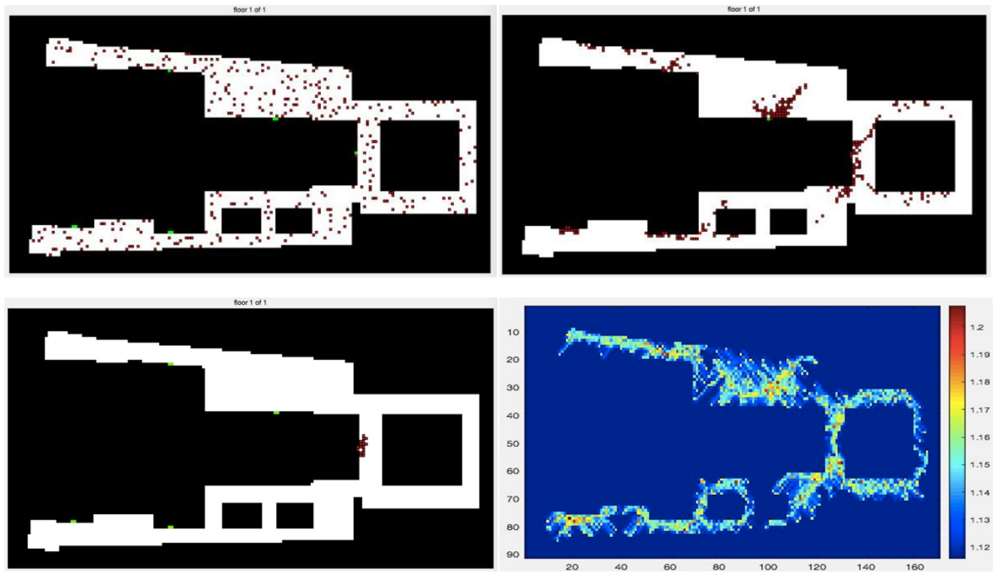


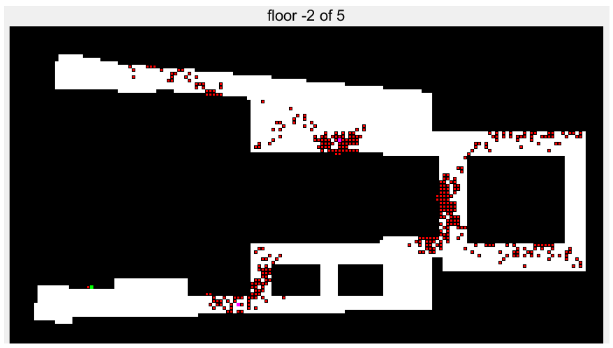
Figure 6: Min number Simulation

The model runs 66 steps to evacuate all the people. From the assumption, the total evacuation time is about 26.4s.

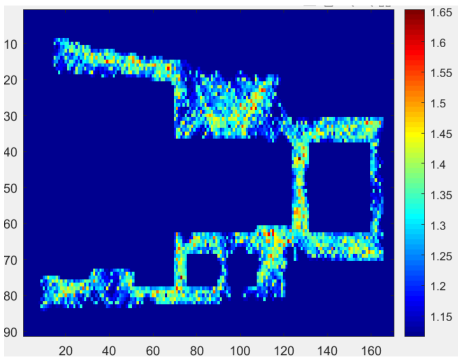
3.2.2 Multi-layer Result

There are 5 floors in the Louvre, so we connect our simple single-layer models with stairs to get the final actual multi-layer model. The positions of the stairs are as follows(Assume that all floors except the 0^th floor):

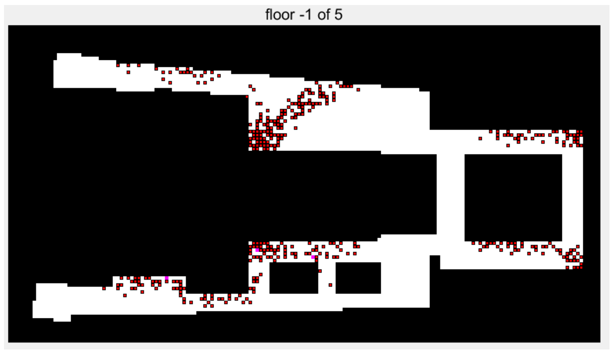
For stairs we use a queue to transfer visitors between stairways on different floors at a certain rate. Eventually, we get that the steps needed to evacuate completely for our multi-layer model are more than five times as many as our single-layer model, due to the congestion at the stairs. That's not the same with the true situation, which will be talked in our model's weakness.



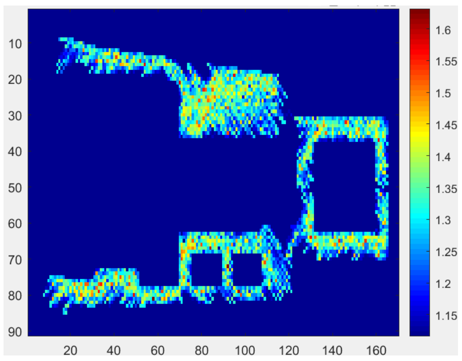
(a)



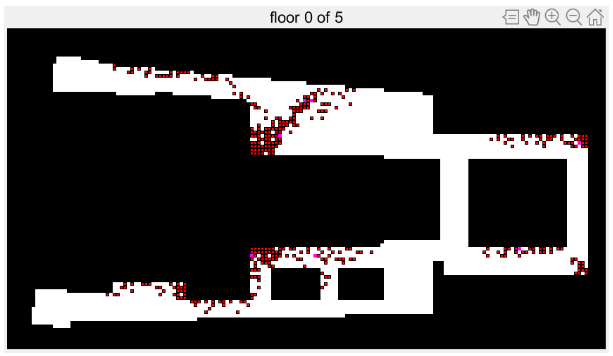
(b)



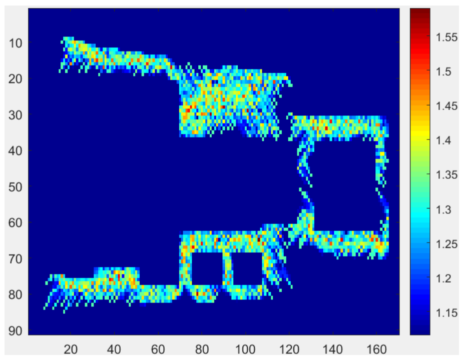
(c)



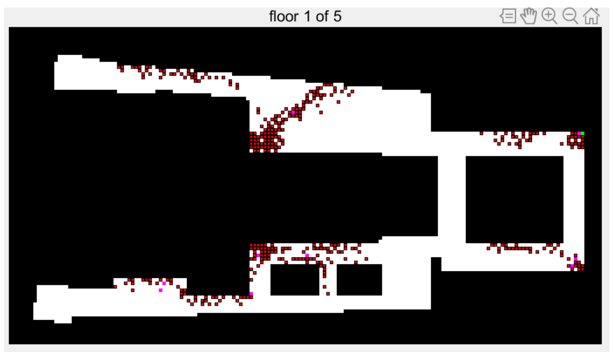
(d)



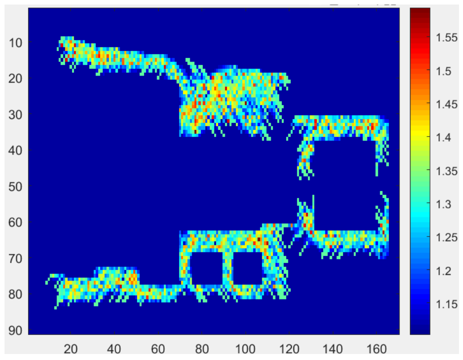
(e)



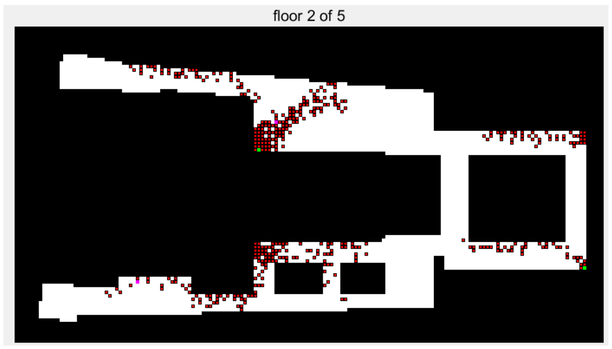
(f)



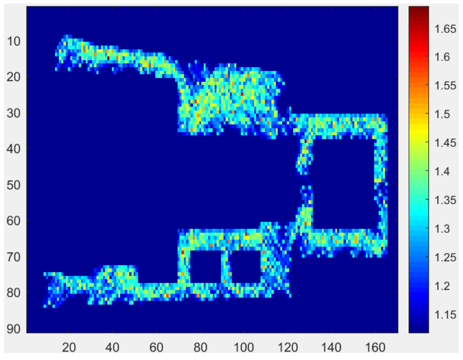
(g)



(h)



(i)



(j)

Figure 7: Multi-Stairs Simulation

3.2.3 Effect of Pheromone

We should focus on the pheromone distribution map besides the time to get out of the Louvre, because we can get much info from it. On the one hand, pheromone tells us the number of people passing through the process of finding exports. The more people pass, the more likely it is the best path leading to the exit. It can be proved when there are not that many people in the model after iteration. On the other hand, the level of congestion can be reflected by the pheromone distribution. Due to the iteration formula of pheromone given by the improved ant colony algorithm and the penalty mechanism introduced in section 3.1.3, the pheromone on the exact path will be significantly reduced when congested. This theory meets our observation that the concentration of pheromone is extremely low around some crowded entrances.

The pheromone can reflect the number of people and congestion, so that we can find the evacuation **bottleneck** and take advantage of **extra emergency exits** accordingly. There are two possibilities when the concentration of pheromone is small: congestion or few passes. We talked about congestion last graph and "few passes" indicates that the path is too far from the door, both of which limit the total evacuation time.

3.3 Extension of the Model

The diversity of tourists is an important part for the real system, so we take different visitors into consideration. And we need to design a route to help emergency personnel enter the building as quickly as possible.

3.3.1 Different Nationality

Visitors from different countries speak a variety of languages in the Louvre, the ratio of different countries is displayed in Figure4 (2)

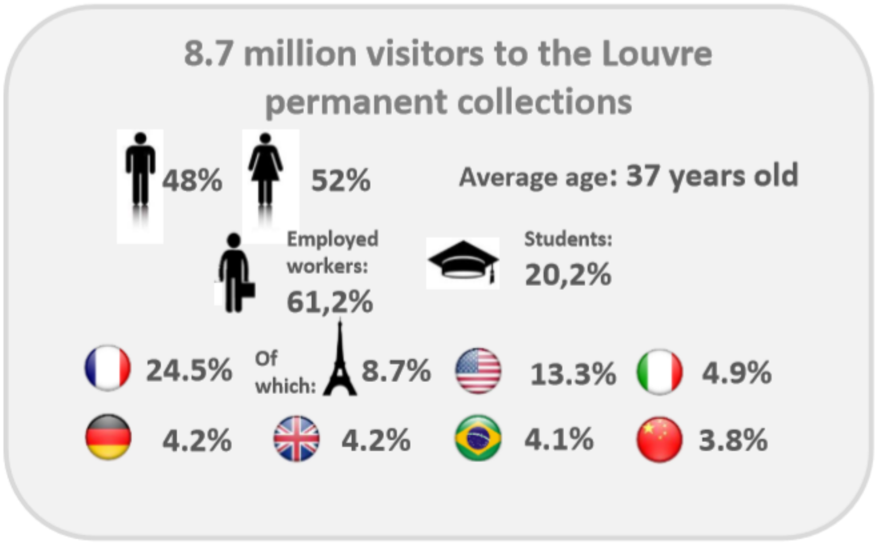
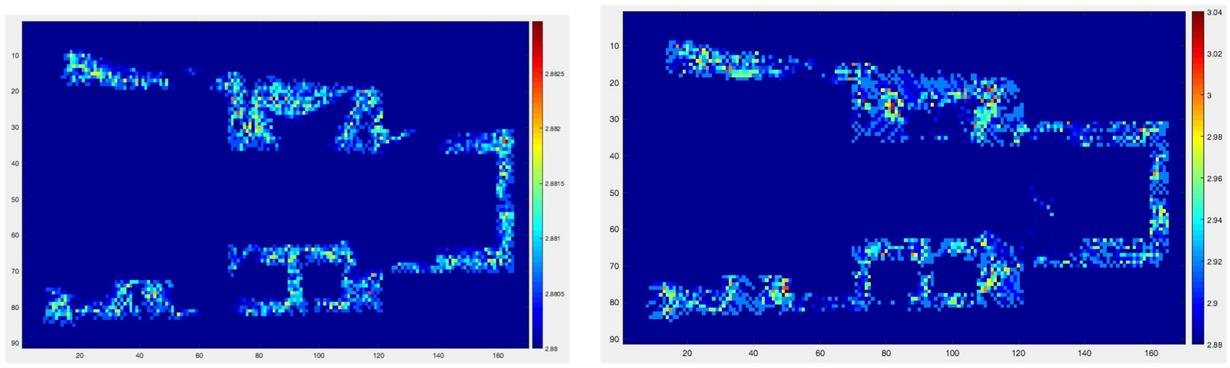


Figure 8: visitors speaking different language

We can see that French and American visitors are the most, so we can use English and French to give instructions. For those who can't understand, they would move with other people.

From our model, if all people move randomly or just directly to the nearest door, the exit will be more congested. Therefore, if we can guide most visitors evacuate following instructions, those who don't know what we say will still go with other people, which can help reduce congestion at one specific door.



(a) random (b) optimized

Figure 9: Contrast in different Situations

Through the pheromone distribution in the process showed in figure(9), we see that random state makes pheromone around the pyramid entrance lower than the optimized state, from which we know that the simulation satisfies our assumption. Thus we don't need to worry much about people speaking different languages, because they will follow others who know French or English.

3.3.2 Disabled Visitors

In our model, people with disabilities can be randomly added to the museum at a certain position (blue cells in the picture), and to match the real situation, they move at a quarter of the speed of other visitors in the model.

In the figure, disabled people can't get out quickly if they are in the right part of building. Thus, we suggest adding exit for disabled here or open near emergency exit.

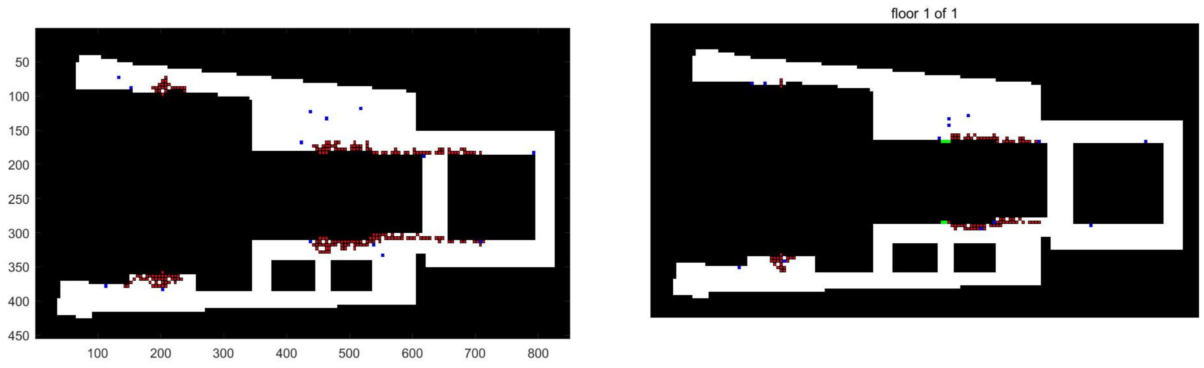


Figure 10: Simulation with Disabled

3.3.3 Groups Traveling Together

In the algorithm, we add two groups of visitors(showed in Figure()), and it can be seen that the appearance of group visitors has no effect on the choice of path. Group visitors may move together or separately. However, participation of group visitors moving together leads to more serious congestion, which is similar with the reality.

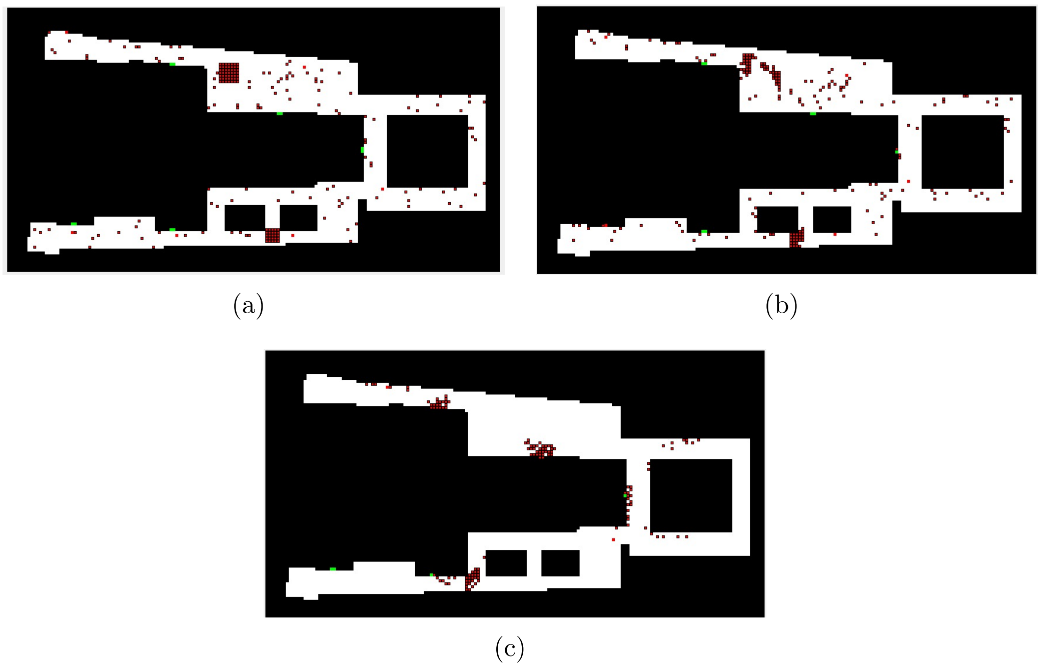


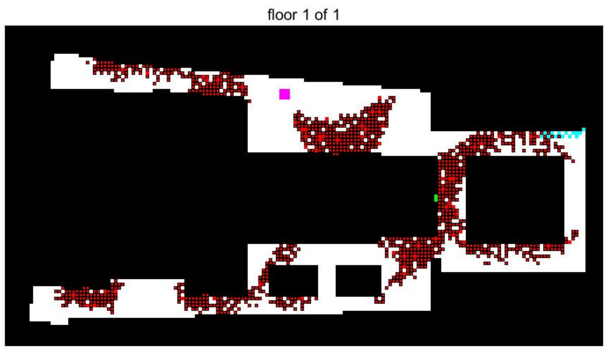
Figure 11: Simulation with Group Visitors

3.3.4 Emergency Personnel

In our model, the heuristic function is given by Dijkstra algorithm.However, it just give the distance, without the path to the door. So we designed A* algorithm for emergency personnel so that they can find a best route to reach where accidents happens.

$$\begin{cases} h(n) = \frac{1}{|X_e - X_a| + |Y_e - Y_a|} \\ g(n) = n \\ f(n) = g(n) + h(n) \end{cases} \tag{7}$$

Where f is the heuristic function required for the n^{th} step, g is the actual path cost, and h uses the Manhattan distance between the emergency personnel and the accident location. Emergency personnel cannot cross obstacles, so this is a admissible heuristic function, and the optimal solution is given. We can see the blue route on the Figure, which represents that the emergency personnel can avoid evacuating people and arrive the place as soon as possible.



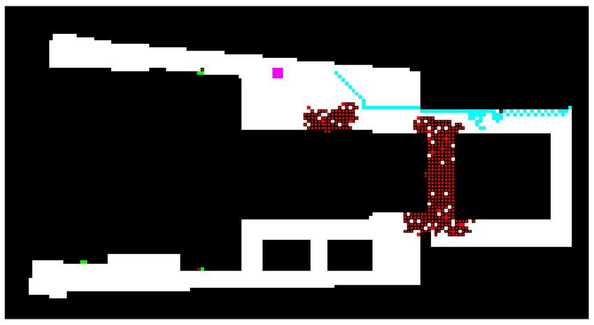


Figure 12: Simulation for Emergency Personnel

4 Anaysis of the Model

4.1 Sensitivity

In this section we mainly analyze thesensitivity of influence factor of pheromone ofthe evacuation model. We analyzed the experimental results of the influence factor of pheromone over a wide range of changes, and obtained the variation curve of people failed to evacuate in 250 stepsas follows:

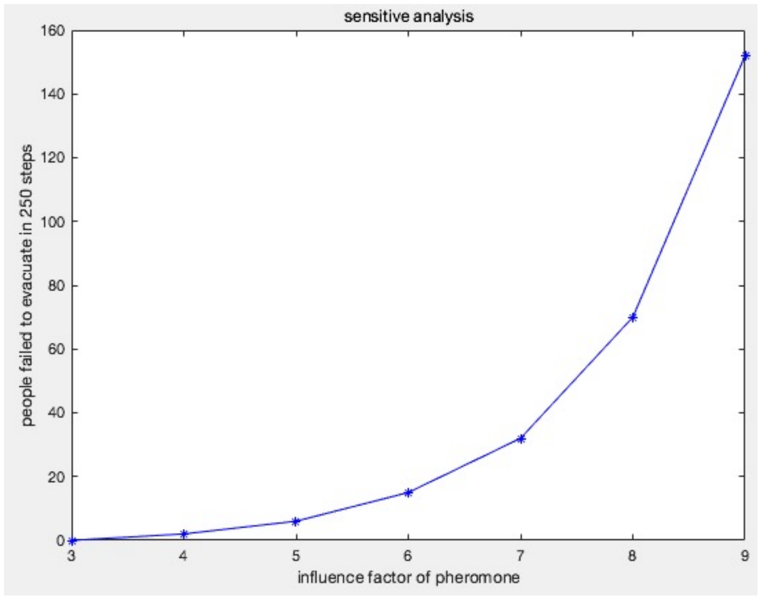


Figure 13: Sensitivity Analysis

As can be seen from the above figure, the sensitivity of influence factor of pheromoneis approximately exponential. This means that when the influence factor of pheromone changes beyond a certain range, the stability of the model will go worse, thus we should be careful when we adjust the influence factors.

4.2 Strengths and Weakness

4.2.1 Strengths

In our model, we make necessary simplification so we can not only run the code fast to get the solution as soon as possible in case of emergency, but also have the flexibility to modify our model to suit different situations, such as fire or terrorist attacks.

The number of input parameters is relatively small and easy to adjust in the proper range, leading to relatively good robustness.

Our model improves the traditional ant colony algorithm, avoids infinite loops, speeds up the convergence, and achieves better evacuation path planning.

4.2.2 Weakness

We have to admit that our simplification of the structure of the Louvre, such all the exits are on the same floor, simplifying the pyramid's architectural structure, etc., actually reduces the accuracy of our model.

In our model, special exit for disabled is not considered. Also, we don't use available exit considering that we don't know the exact position of these exits and the exits have a large influence on our model.

We don't know enough about the real distribution of the different rooms in the Louvre and can only be approximated by random distribution.But at the same time museum administrators can use our code to set limitation of the density of visitors to get more accurate evacuation advice.

5 Further Discussion

5.1 Assistance by App

The app Affluences provides real-time updates on the estimated waiting time at each of these entrances to help facilitate entry to the museum so that we can use it to estimate the level of congestion at the exit.

For the reason that the entrances and exits are generally together, when an accident occurs, the entrance can be used as an exit after the person waiting in line at the entrance are evacuated, which is equivalent to widening the exit in order to reduce the congestion of the exit. The time to evacuate people at the entrance can be taken as the time to get into the museum from this entrance approximately, which we can get from Affluences. Then visitors with this app can get the information and can move to the door which is not crowded.

5.2 Optimization for Stairs

In the three-dimensional analysis, our current model has no restrictions on the length of the stairs. the number of people at stairs can always increase without speed loss.But in the real time, there is capacity restriction on the stairs, that means when the number of people in the stairs reaches the limit, it can't accommodate more people. This can be described using Queuing Theory.

Assume that from time t, the time of arrival of the next person is distributed with a negative exponential probability of the parameter λ. From the time t, the time of the next person leaving is distributed with a negative exponential probability of the parameter μ.

$$C_n = \frac{\lambda_{n-1}\lambda_{n-2}...\lambda_0}{\mu_n\mu_{n-1}...\mu_1}, n = 1, 2, 3... \tag{8}$$

if $\sum_{n=1}^{\infty} C_n$ is convergent, the length of the queue is convergent. Then we can use the theory to solve problem like this, especially larger museums or with more complicated stairs.

5.3 Comparison with Simulation Software

Comparison with the commercial software Pathfinder: We use Pathfinder to simulate the evacuation process in the first floor in the Louvre, and we get the result as follows:

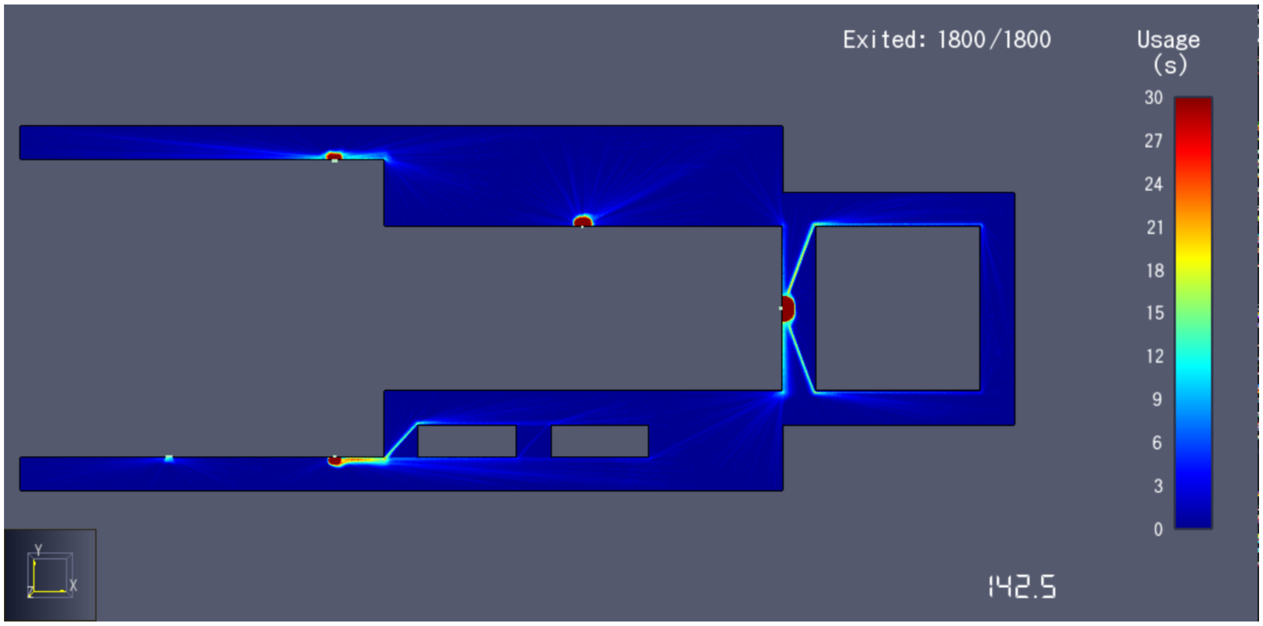


Figure 14: simulation by pathfinder

It takes 142.5 seconds for 1800 people to evacuate from the Louvre. The main escape routes can be obtained by the total time of visitors using a certain path during the evacuation process, which is the brighter area in the above picture. It can also be seen from the figure that the exit on the right is the most crowded, and the exit rate in the lower left corner is the lowest. This is consistent with the main escape routes based on our model.
In our model, evacuation of 1800 people requires a total of 380 steps. In the Moore cellular automaton, the distance of one step is about $2m$, and in an emergency, the speed of the visitor is about $5m/s$ (when not in the congested area). So it takes $0.4s$ to move one step and $152s$ in 380 steps. Compared with the simulation results of Pathfinder, the time taken

is longer, but the difference is not very large.

What's more, compared with the simulation software, our model not only considers evacuating tourists, but also considers the situation that emergency personnel enter the museum at the same time, and optimally plans their path, which cannot be done by simulation software.

6 Conclusion

Our model can help the Louvre administrator manage emergency situationsin three ways:
1. Flexible evacuationmodel: Through the built model, the Louvre administrator can simulate the basic evacuation model using the average density of the visitors and get the suggestions of general evacuation directions in the Louvre. Most importantly, he/she can get the congestion info from the pheromone distribution, aware of the **bottleneck**, open some emergency exits in purpose, and also do some restrictions on number of people in some rooms regularly.

2. Accurate evacuation model: During the simulation process in our model, the Louvre administrator could obtain evacuation directions suggested for different positions in the Louvre in real time. If this information could be sent bythe Louvre apptoeveryvisitor, theneach visitor's evacuation route is accurately planned according to the model.

3. Model dealing with different risks: the Louvre administrator can set the entrance used by emergency personnel and the kind of emergency in the model, thus the algorithm has strong adaptive ability and can adapt to different types of risks. In the case of a fire, for example, the model can be used to simulatethe evacuation routesfor tourists to avoid fire sources and provide the best fire-fighting route for rescuers.

After increasing the number of visitors in the Louvre, we can see that the model still works well. Therefore, for other large, crowded buildings, our model can also be used directly. At the same time, the simulation map in the model can be automatically generated by the image imported, so it has a strong generalization ability and is practical for buildings with different shapes.

References

[1] 8.1 Million Visitors to the Louvre in 2017. Louvre Press Release, 25 Jan. 2018.

[2] Pyramid Project Launch The Muse du Louvre is improving visitor reception (2014-2016). Louvre Press Kit, 18 Sept. 2014.

[3] WANG Ru, ZHOU Lei and LIU Jun, Study on cellular automaton evacuation model based on improved ant colony optimization algorithm, China Safety Science Journal ,Jan 2018.

[4] Fu Jundong, Liu Yehui, Li Jianghui. Dynamic Discharge of Fire Based on Ant Colony Algorithm[J]. Journal of East China Jiaotong University, 2017(3).

[5] Ji-Hong S, Kan W, Pu LI. The Research of Passenger's Evacuation in Ships by Using Improved Ant Colony Algorithm[J]. Control Engineering of China, 2013.

[6] Chunshan L U , Wenguo W , Rui Y , et al. Fire evacuation model based on motor schema and cellular automaton[J]. Journal of Tsinghua University, 2007, 47(12):2163-2167.

[7] Fang Z, Zong X, Li Q, et al. Hierarchical multi-objective evacuation routing in stadium using ant colony optimization approach[J]. Journal of Transport Geography, 2011, 19(3):443-451.

[8] Sabri N A M , Basari A S H , Hussin B , et al. Ant colony-dijkstras algorithm for evacuation preparedness in high rise buildings[M]. Advanced Computer and Communication Engineering Technology. Springer International Publishing, 2016.

[9] Liu M, Zhang F, Ma Y, et al. Evacuation path optimization based on quantum ant colony algorithm[J]. Advanced Engineering Informatics, 2016, 30(3):259-267.

[10] Dorigo M, Gambardella L M, Birattari M, et al. Ant Colony Optimization and Swarm Intelligence[M]// Ant colony optimization and swarm intelligence. 2004.

[11] BLUM, Christian. Ant colony optimization : Introduction and recent trends[J]. Physics of Life Reviews, 2005, 2(4):353-373.

A Appendix1 Matlab Code

```
classdef cellmachine < handle

properties
cellmap;
M;
N;

peoplenum_total;
peoplenum_now;

people_position;
start_position;
disable_position;
door_position;
wall_position;
target=[20,80;21,80;
19,80;20,81;20,79;
19,79;19,81;21,79;21,81];
start_door=[30,165];
security_position;
path;
Lk;
count;

ro=0.1;
epoch=1;

arrived_flag = 0;

end

methods
function obj =
cellmachine(varargin)

if nargin==1
map=
varargin{1};
obj.start_position=[];
obj.people_position=[];
obj.door_position=[];
obj.wall_position=[];
obj.disable_position=[];
for i=1:size(map,1)
for j=1:size(map,2)
if(map(i,j)==3)
     obj.door_position=
     [obj.door_position;[i,j]];
end
if(map(i,j)==0)
     obj.wall_position=
     [obj.wall_position;[i,j]];
end
end
end

obj.M=size(map,1);
obj.N=size(map,2);

peoplenum=4;
N=randperm(obj.M*obj.N,
ceil(obj.M*obj.N/peoplenum));
end

if  nargin==0
obj.start_position=[3,3;
     5,5;4,4;6,6;7,7;8,8;3,4;4,5;
     5,6;6,7;7,8;8,9;3,9;4,8;5,7;
     7,5;8,4;9,3;9,6;2,9];
obj.people_position=[3,3;5,5;4,4;
     6,6;7,7;8,8;3,4;4,5;5,6;6,7;
     7,8;8,9;3,9;4,8;5,7;7,5;8,4;
     9,3;9,6;2,9];
obj.door_position=[10,8;1,2];
obj.wall_position=[2,2;3,2;4,2;
     5,2];
obj.M=10;
obj.N=10;
end

for i=1:obj.M
for j=1:obj.N
obj.cellmap{i,j}=mycell(i,j,
0,3*obj.M+3*obj.N,0.5,0.5);
end
end
```

```
for i=1:size(obj.people_position,1)
obj.cellmap{obj.people_position(i,1),
obj.people_position(i,2)}.category=1;
end
for i=1:size(obj.door_position,1)
obj.cellmap{obj.door_position(i,1),
obj.door_position(i,2)}.category=3;
end
for i=1:size(obj.wall_position,1)
obj.cellmap{obj.wall_position(i,1),
obj.wall_position(i,2)}.category=2;
end

for i=1:size(obj.target,1)
obj.cellmap{obj.target(i,1),
obj.target(i,2)}.category=4;
end
obj.cellmap{obj.start_door(1),
obj.start_door(2)}.category=5;

obj.security_position = obj.start_door;

for k=1:obj.M*obj.N/peoplenum
temp_n=mod(N(k),obj.N);
if temp_n==0
temp_n=obj.N;
end
if obj.cellmap{ceil(N(k)/obj.N),
     temp_n}.category==0
obj.people_position=[obj.people_position;
     [ceil(N(k)/obj.N),temp_n]];
obj.start_position=[obj.start_position;
     [ceil(N(k)/obj.N),temp_n]];

if mod(k,20) == 0
obj.cellmap{ceil(N(k)/obj.N),
temp_n}.category=6;
obj.disable_position=
[obj.disable_position;[ceil(N(k)/obj.N),temp_n]];
else
obj.cellmap{ceil(N(k)/obj.N),
temp_n}.category=1;
end
```

```
end

end


for i=1:size(obj.people_position ,1)
obj.Lk(i)=1;
obj.path{i} = [obj.people_position(i,1),
    obj.people_position(i,2)];
end
obj.peoplenum_total=size(obj.people_position ,1);
obj.peoplenum_now=size(obj.people_position ,1);
obj.count =
zeros(obj.peoplenum_total,obj.M,obj.N,8);
calculate(obj);

end

function one_step(obj,first_flag ,step_flag)
for i=1:size(obj.people_position ,1)
if obj.people_position(i,1)~=-1
obj.Lk(i)=obj.Lk(i)+1;
end
end
if first_flag == obj.epoch &&
    obj.arrived_flag == 0
present_cell =
obj.cellmap{obj.security_position(1),
obj.security_position(2)};
next_cell = present_cell;

if present_cell.parent(1) ~= 0
if obj.cellmap{present_cell.parent(1),
    present_cell.parent(2)}.category == 0 ||
    obj.cellmap{present_cell.parent(1),
    present_cell.parent(2)}.category == 4
next_cell =
obj.cellmap{present_cell.parent(1),
present_cell.parent(2)};
else
distance = [];
for i=-1:1
for j=-1:1
new_distance =
abs(obj.security_position(1)+i-obj.target(1,1))+
abs(obj.security_position(2)+j-obj.target(1,2));
distance=[distance,new_distance];
end
end
[~,num] = sort(distance);

for j = 1:9
    if num(j) == 1 &&
        obj.security_position(1)-1>0 &&
        obj.security_position(2)-1>0
        next_cell =
        obj.cellmap{obj.security_position(1)-1,
        obj.security_position(2)-1};
    end
    if num(j) == 2 && obj.security_position(1)-1>0 &&
        obj.security_position(2)<=obj.N
        next_cell =
        obj.cellmap{obj.security_position(1)-1,
        obj.security_position(2)};
    end
    if num(j) == 3 && obj.security_position(1)-1>0 &&
        obj.security_position(2)+1<=obj.N
        next_cell =
        obj.cellmap{obj.security_position(1)-1,
        obj.security_position(2)+1};
    end
    if num(j) == 4 && obj.security_position(1)>0 &&
        obj.security_position(2)-1>=obj.N
        next_cell =
        obj.cellmap{obj.security_position(1),
        obj.security_position(2)-1};
    end

    if num(j) == 6 &&
        obj.security_position(1)<=obj.M &&
        obj.security_position(2)+1<=obj.N
        next_cell =
        obj.cellmap{obj.security_position(1),
        obj.security_position(2)+1};
    end
    if num(j) == 7 &&
        obj.security_position(1)+1<=obj.M &&
        obj.security_position(2)-1>0
```

```
            next_cell =
            obj.cellmap{obj.security_position(1)+1,
            obj.security_position(2)-1};
        end
        if num(j) == 8 &&
            obj.security_position(1)+1<=obj.M &&
            obj.security_position(2)>0
            next_cell =
            obj.cellmap{obj.security_position(1)+1,
            obj.security_position(2)};
        end
        if num(j) == 9 &&
            obj.security_position(1)+1<=obj.M &&
            obj.security_position(2)+1<=obj.N
            next_cell =
            obj.cellmap{obj.security_position(1)+1,
            obj.security_position(2)+1};
        end

        if next_cell.category == 0 ||
            next_cell.category == 4

            break;
        end
end
end

else
distance = [];
for i=-1:1
for j=-1:1
    new_distance =
    abs(obj.security_position(1)+i-obj.target(1,1))+
    abs(obj.security_position(2)+j-obj.target(1,2));
    distance=[distance,new_distance];
end
end
[~,num] = sort(distance);

for j = 1:9
if num(j) == 1 &&
    obj.security_position(1)-1>0 &&
    obj.security_position(2)-1>0
    next_cell =
    obj.cellmap{obj.security_position(1)-1,
    obj.security_position(2)-1};
end
if num(j) == 2 &&
    obj.security_position(1)-1>0 &&
    obj.security_position(2)<=obj.N
    next_cell =
    obj.cellmap{obj.security_position(1)-1,
    obj.security_position(2)};
end
if num(j) == 3 &&
    obj.security_position(1)-1>0 &&
    obj.security_position(2)+1<=obj.N
    next_cell =
    obj.cellmap{obj.security_position(1)-1,
    obj.security_position(2)+1};
end
if num(j) == 4 &&
    obj.security_position(1)>0 &&
    obj.security_position(2)-1>=obj.N
    next_cell =
    obj.cellmap{obj.security_position(1),
    obj.security_position(2)-1};
end

if num(j) == 6 &&
    obj.security_position(1)<=obj.M &&
    obj.security_position(2)+1<=obj.N
    next_cell =
    obj.cellmap{obj.security_position(1),
    obj.security_position(2)+1};
end
if num(j) == 7 &&
    obj.security_position(1)+1<=obj.M &&
    obj.security_position(2)-1>0
    next_cell =
    obj.cellmap{obj.security_position(1)+1,
    obj.security_position(2)-1};
end
if num(j) == 8 &&
    obj.security_position(1)+1<=obj.M &&
    obj.security_position(2)>0
    next_cell =
    obj.cellmap{obj.security_position(1)+1,
    obj.security_position(2)};
end
if num(j) == 9 &&
    obj.security_position(1)+1<=obj.M &&
    obj.security_position(2)+1<=obj.N
    next_cell =
    obj.cellmap{obj.security_position(1)+1,
    obj.security_position(2)+1};
end

if next_cell.category == 0 ||
    next_cell.category == 4
    break;
end
end
end

if next_cell.category == 4
step_flag
obj.arrived_flag = 1;
else
next_cell.category = 5;
obj.cellmap{obj.security_position(1),
obj.security_position(2)}.category = 5;
obj.security_position(1) = next_cell.x;
obj.security_position(2) = next_cell.y;
end
end

for i = 1:size(obj.people_position,1)

if obj.people_position(i,1)==-1
continue;
end

cost=obj.M*obj.N*ones(8,1);
if obj.people_position(i,1)-1>0 &&
    obj.people_position(i,2)-1>0
cost(1)=obj.cellmap{obj.people_position(i,1)-1,
obj.people_position(i,2)-1}.cost;
end
if obj.people_position(i,1)-1>0
cost(2)=obj.cellmap{obj.people_position(i,1)-1,
obj.people_position(i,2)}.cost;
```

```
end
if obj.people_position(i,1)-1>0 &&
    obj.people_position(i,2)+1<=obj.N
cost(3)=obj.cellmap{obj.people_position(i,1)-1,
obj.people_position(i,2)+1}.cost;
end
if obj.people_position(i,2)-1>0
cost(8)=obj.cellmap{obj.people_position(i,1),
obj.people_position(i,2)-1}.cost;
end
if  obj.people_position(i,2)+1<=obj.N
cost(4)=obj.cellmap{obj.people_position(i,1),
obj.people_position(i,2)+1}.cost;
end
if obj.people_position(i,1)+1<=obj.M &&
    obj.people_position(i,2)-1>0
cost(7)=obj.cellmap{obj.people_position(i,1)+1,
obj.people_position(i,2)-1}.cost;
end
if obj.people_position(i,1)+1<=obj.M
cost(6)=obj.cellmap{obj.people_position(i,1)+1,
obj.people_position(i,2)}.cost;
end
if obj.people_position(i,1)+1<=obj.M &&
    obj.people_position(i,2)+1<=obj.N
cost(5)=obj.cellmap{obj.people_position(i,1)+1,
obj.people_position(i,2)+1}.cost;
end

obj.cellmap{obj.people_position(i,1),
obj.people_position(i,2)}.walk_a_step(cost);
end

for i = 1:size(obj.people_position,1)

if obj.people_position(i,1)==-1
continue;
end

present_cell =
obj.cellmap{obj.people_position(i,1),
obj.people_position(i,2)};
next_cell = present_cell;
change_flag = 0;
```

```
change_position = 0;
if present_cell.category == 6 &&
    present_cell.stay_time ~= 4
present_cell.stay_time =
present_cell.stay_time + 1;
continue;
end
out_flag = 0;
for k=1:size(obj.door_position,1)
if present_cell.x == obj.door_position(k,1) &&
    present_cell.y == obj.door_position(k,2)
obj.people_position(i,1) = -1;
obj.people_position(i,2) = -1;
obj.peoplenum_now = obj.peoplenum_now - 1;
present_cell.category = 3;
out_flag = 1;



break;
end
end

if obj.people_position(i,1) == -1 || out_flag == 1
continue;
end

[~,num] = sort(present_cell.next_step,'descend');

if first_flag == 1 || first_flag ==
    obj.epoch || mod(first_flag,2) ~= 0 ||
    mod(step_flag,4) ~= 0
for j = 1:8
if num(j) == 1 && obj.people_position(i,1)-1>0 &&
    obj.people_position(i,2)-1>0
    next_cell =
    obj.cellmap{obj.people_position(i,1)-1,
    obj.people_position(i,2)-1};
end
if num(j) == 2 && obj.people_position(i,1)-1>0 &&
    obj.people_position(i,2)<=obj.N
    next_cell =
    obj.cellmap{obj.people_position(i,1)-1,
    obj.people_position(i,2)};
```

```
end
if num(j) == 3 && obj.people_position(i,1)-1>0 &&
    obj.people_position(i,2)+1<=obj.N
    next_cell =
    obj.cellmap{obj.people_position(i,1)-1,
    obj.people_position(i,2)+1};
end
if num(j) == 4 && obj.people_position(i,1)>0 &&
    obj.people_position(i,2)+1<=obj.N
    next_cell =
    obj.cellmap{obj.people_position(i,1),
    obj.people_position(i,2)+1};
end
if num(j) == 5 &&
    obj.people_position(i,1)+1<=obj.M &&
    obj.people_position(i,2)+1<=obj.N
    next_cell =
    obj.cellmap{obj.people_position(i,1)+1,
    obj.people_position(i,2)+1};
end
if num(j) == 6 &&
    obj.people_position(i,1)+1<=obj.M &&
    obj.people_position(i,2)<=obj.N
    next_cell =
    obj.cellmap{obj.people_position(i,1)+1,
    obj.people_position(i,2)};
end
if num(j) == 7 &&
    obj.people_position(i,1)+1<=obj.M &&
    obj.people_position(i,2)-1>0
    next_cell =
    obj.cellmap{obj.people_position(i,1)+1,
    obj.people_position(i,2)-1};
end
if num(j) == 8 &&
    obj.people_position(i,1)>0 &&
    obj.people_position(i,2)-1>0
    next_cell =
    obj.cellmap{obj.people_position(i,1),
    obj.people_position(i,2)-1};
end

if next_cell.category == 0 ||
    next_cell.category == 3
    change_flag = 1;
    change_position = num(j);
    break;
end
end

else

A=ones(1,100);
x=rand(1,100);
A(x<present_cell.next_step(1))=1;
sum_posibility = 0;
for j=1:7
sum_posibility = sum_posibility +
present_cell.next_step(j);
A(x>=sum_posibility)=j+1;
end
position_tried = zeros(8,1);
try_step = 0;
while sum(position_tried) < 8 &&
    try_step < 30
result = A(randperm(100,1));
position_tried(result) = 1;
if result == 1 &&
    obj.people_position(i,1)-1>0 &&
    obj.people_position(i,2)-1>0
    next_cell =
    obj.cellmap{obj.people_position(i,1)-1,
    obj.people_position(i,2)-1};
end
if result == 2 &&
    obj.people_position(i,1)-1>0 &&
    obj.people_position(i,2)<=obj.N
    next_cell =
    obj.cellmap{obj.people_position(i,1)-1,
    obj.people_position(i,2)};
end
if result == 3 &&
    obj.people_position(i,1)-1>0 &&
    obj.people_position(i,2)+1<=obj.N
    next_cell =
    obj.cellmap{obj.people_position(i,1)-1,
    obj.people_position(i,2)+1};
end
```

```
if result == 4 &&
    obj.people_position(i,1)>0 &&
    obj.people_position(i,2)+1<=obj.N
    next_cell =
    obj.cellmap{obj.people_position(i,1),
    obj.people_position(i,2)+1};
end
if result == 5 &&
    obj.people_position(i,1)+1<=obj.M &&
    obj.people_position(i,2)+1<=obj.N
    next_cell =
    obj.cellmap{obj.people_position(i,1)+1,
    obj.people_position(i,2)+1};
end
if result == 6 &&
    obj.people_position(i,1)+1<=obj.M &&
    obj.people_position(i,2)<=obj.N
    next_cell =
    obj.cellmap{obj.people_position(i,1)+1,
    obj.people_position(i,2)};
end
if result == 7 &&
    obj.people_position(i,1)+1<=obj.M &&
    obj.people_position(i,2)-1>0
    next_cell =
    obj.cellmap{obj.people_position(i,1)+1,
    obj.people_position(i,2)-1};
end
if result == 8 &&
    obj.people_position(i,1)>0 &&
    obj.people_position(i,2)-1>0
    next_cell =
    obj.cellmap{obj.people_position(i,1),
    obj.people_position(i,2)-1};
end

if next_cell.category == 0 ||
    next_cell.category == 3
    change_flag = 1;
    change_position = result;
    break;
end
try_step = try_step + 1;
end
```

```
end
if change_flag == 1
if present_cell.category == 1
next_cell.category = 1;
end
if present_cell.category == 6
next_cell.category = 6;
end
present_cell.category = 0;
present_cell.stay_time = 0;
obj.people_position(i,1) = next_cell.x;
obj.people_position(i,2) = next_cell.y;
obj.path{i} = [obj.path{i};[next_cell.x,next_cell.y]];

obj.count(k,present_cell.x,present_cell.y,change_position) = obj.count(k,pres
end
end
end

function [time_05,time_075]=
     one_iteration(obj,time)
step = 0;
max_step = 250;
time_05=0;
time_075=0;
while step < max_step
if obj.peoplenum_now == 0
step
break;
end
drawmap(obj,1,1)
obj.one_step(time,step);
step = step + 1;

if obj.peoplenum_now<
     (0.5*obj.peoplenum_total)
if time_05==0
time_05=step;
end
end
if obj.peoplenum_now<
     (0.25*obj.peoplenum_total)
if time_075==0
```

```
time_075=step;
end
end
end

for i=1:size(obj.cellmap,1)
for k=1:size(obj.cellmap,2)
for j=1:8
obj.cellmap{i,k}.info(j)=
(1-obj.ro) * obj.cellmap{i,k}.info(j);

end
end
end

visited = zeros(obj.M,obj.N,obj.M,obj.N);
for i=1:size(obj.people_position,1)

punish_flag = 0;
for j=size(obj.path{i},1)-1:-1:1
if obj.path{i}(j,1) ==
    obj.path{i}(j+1,1) &&
    obj.path{i}(j,2) == obj.path{i}(j+1,2)
continue;
end
if visited(obj.path{i}(j,1),
    obj.path{i}(j,2),obj.path{i}(j+1,1),
    obj.path{i}(j+1,2)) == 1
continue;
end
num = 0;
if obj.path{i}(j,1)-1 == obj.path{i}(j+1,1)
if obj.path{i}(j,2)-1 == obj.path{i}(j+1,2)
    num = 1;
end
if obj.path{i}(j,2) == obj.path{i}(j+1,2)
    num = 2;
end
if obj.path{i}(j,2)+1 == obj.path{i}(j+1,2)
    num = 3;
end
end
if obj.path{i}(j,1) == obj.path{i}(j+1,1)
if obj.path{i}(j,2)-1 == obj.path{i}(j+1,2)
```

```
    num = 8;
end
if obj.path{i}(j,2) == obj.path{i}(j+1,2)
    num = 0;
end
if obj.path{i}(j,2)+1 == obj.path{i}(j+1,2)
    num = 4;
end
end
if obj.path{i}(j,1)+1 == obj.path{i}(j+1,1)
if obj.path{i}(j,2)-1 == obj.path{i}(j+1,2)
    num = 7;
end
if obj.path{i}(j,2) == obj.path{i}(j+1,2)
    num = 6;
end
if obj.path{i}(j,2)+1 == obj.path{i}(j+1,2)
    num = 5;
end
end

if  obj.count(i,obj.path{i}(j,1),
    obj.path{i}(j,2),num) >= 5 ||
    punish_flag == 1



obj.cellmap{obj.path{i}(j,1),
obj.path{i}(j,2)}.info(num) =
obj.cellmap{obj.path{i}(j,1),
obj.path{i}(j,2)}.info(num) * (1-obj.ro)^2;
punish_flag = 1;
continue;
end

delta_info = 0;
for k=1:size(obj.people_position,1)

num_reverse = mod(num+4,8);
if num_reverse == 0
    num_reverse = 8;
end
n = obj.count(k,obj.path{i}(j,1),
obj.path{i}(j,2),num) +
obj.count(k,obj.path{i}(j+1,1),
obj.path{i}(j+1,2),num_reverse);

if n ~= 0
    delta_info = delta_info +
    1/obj.Lk(k)/obj.Lk(k)*0.5^(n-1);
end
end
visited(obj.path{i}(j,1),obj.path{i}(j,2),
obj.path{i}(j+1,1),obj.path{i}(j+1,2)) = 1;
visited(obj.path{i}(j+1,1),
obj.path{i}(j+1,2),obj.path{i}(j,1),
obj.path{i}(j,2)) = 1;
present_cell =
obj.cellmap{obj.path{i}(j,1),
obj.path{i}(j,2)};
present_cell.info(num) =
present_cell.info(num) + delta_info;
end

if obj.cellmap{obj.path{i}(end,1),
    obj.path{i}(end,2)}.category == 1 ||
    obj.cellmap{obj.path{i}(end,1),
    obj.path{i}(end,2)}.category == 6

obj.cellmap{obj.path{i}(end,1),
obj.path{i}(end,2)}.category = 0;
end

end

end

function run(obj)


time_05=[];
time_075=[];
for i = 1:obj.epoch
i
for j=1:size(obj.people_position,1)
obj.Lk(j)=1;
end
```

```
if i == obj.epoch
obj.cellmap{obj.start_door(1),
obj.start_door(2)}.category = 5;
obj.security_position =
obj.start_door;
[~,~]=Astar(obj.cellmap)
end

[t05,t075]=obj.one_iteration(i);
time_05=[time_05,t05];
time_075=[time_075,t075];
obj.path = {};
for j=1:size(obj.people_position,1)
obj.path{j} =
[obj.start_position(j,1),
    obj.start_position(j,2)];
for k=1:size(obj.disable_position,1)
if obj.start_position(j,1) ==
    obj.disable_position(k,1) &&
    obj.start_position(j,2) ==
    obj.disable_position(k,2)
    obj.cellmap{obj.start_position(j,1),
    obj.start_position(j,2)}.category = 6;
else
    obj.cellmap{obj.start_position(j,1),
    obj.start_position(j,2)}.category = 1;
end
end

end
obj.people_position = obj.start_position;
obj.peoplenum_now = obj.peoplenum_total;
obj.count =
zeros(obj.peoplenum_total,obj.M,obj.N,8);

end

for i=1:obj.M
for j=1:obj.N
cellinfo(i,j)=sum(obj.cellmap{i,j}.info(:));
end
end
save('info','cellinfo');
x=1:obj.epoch;
```

```
figure(2);
plot(x,time_05,'-*b',x,time_075,'-or');
end
end
end

classdef mycell < handle
properties
x;
y;

category;
next_step;

info;
info_using;

cost;
stay_time;

alfa;
beta;

parent=[0,0];
end

methods
function obj = mycell(startx,starty,
    category_input,cost_input,a,b)
obj.x = startx;
obj.y = starty;
obj.category = category_input;
obj.cost = cost_input;
obj.alfa = a;
obj.beta = b;
obj.info=0.4*ones(8,1);
obj.info_using = obj.info;
obj.stay_time = 0;

end

function walk_a_step(obj,cost_input)
sum_cost = 0;
for i=1:8
```

```
sum_cost = sum_cost +
obj.info(i)^obj.alfa *
(1/(cost_input(i)+0.1))^obj.beta;
end

for i=1:8
obj.next_step(i) = (obj.info(i)^obj.alfa *
(1/(cost_input(i)+0.1))^obj.beta) / sum_cost;
end

end
end
end
```