



The University of Hong Kong

Faculty of Engineering

Department of Computer Science

COMP7705

Project Report

Travelling Route Generation System

Submitted in partial fulfillment of the requirements for the admission to
the degree of Master of Science in Computer Science

By

Li Zhihui (3035562930), Liu Ziyi (3035561845), Zhang Yong (3035561704)

Supervisor: Dr. T.W. Chim

Date of submission: [28/07/2019]

Abstract

With the development of the economy, people's demand for tourism is growing. However, it is very difficult to plan a reasonable, economic and specific travel route in a short period. So, this project offers a convenient and fast way to plan the travel route for the user. The user only needs to input multiple destinations and the total travelling time and then the system recommends the travel route as best as possible based on the comprehensive consideration of the travel route cost. In addition, the system will also plan the detailed route between the travel spots in the city and display an intuitive view of travel spots. And the route is reasonably connected by several travel spots or travel cities by the traffic route. This project collected from Ctrip.com, Qunar.com and Baidu Map API. This report will discuss the algorithm of the route generation and the advantages and disadvantages of the final system compared to other products.

Declaration

We declare that this report is our own unaided work. It is being submitted in partial fulfilment of the degree of Master of Science in Computer Science to the University of Hong Kong. It has not been submitted before for any degree or examination to any other University.

Li Zhihui (3035562930), Liu Ziyi (3035561845), Zhang Yong (3035561704)

28/07/2019

Acknowledgements

We would like to express our deep gratitude to Dr. T.W. Chim, our project supervisors, for his patient guidance, enthusiastic encouragement and useful critiques of this project work. We would also like to thank Priscilla Lam and other staffs for their advice and assistance in keeping our progress on schedule. Finally, we wish to thank our parents for their support and encouragement throughout our study.

Table of Contents

1	Introduction.....	1
1.1	The subject matter and the scope of the investigation	1
1.2	The purpose of the project	2
1.3	The organization of the report	3
1.4	Survey of previously published work and current trends	4
2	Analysis of problem	6
2.1	Data collection.....	6
2.2	Simplification of the problem.....	7
3	Methodology	7
3.1	Data collection.....	7
3.1.1	Web crawling	8
3.1.2	Baidu Map API	9
3.2	Itinerary generation.....	10
3.2.1	Terminology definition	10
3.2.2	Planning algorithm.....	10
3.2.3	Solutions for TSP problem.....	14
3.2.4	Intra-city Route	17
3.2.5	Inter-city Route	18
4	Design and construction of software system.....	21
4.1	Use case modeling.....	21
4.2	Module Interactions.....	22
4.3	Architecture Modeling.....	23
5	Algorithmic/Experimental results	25
5.1	Web Crawler subsystem	25
5.2	Web Server subsystem.....	28
6	Analysis of results.....	35
6.1	Analysis of data collection results	35
6.1.1	Advantages.....	35
6.1.2	Disadvantages	36
6.2	Analysis of web crawling results.....	36
6.2.1	Advantages.....	36

6.2.2 Disadvantages	36
6.3 Analysis of map API results	37
6.3.1 Advantages.....	37
6.3.2 Disadvantages	38
6.4 Analysis of website development results.....	39
6.4.1 Advantages.....	39
6.4.2 Disadvantages	39
6.5 Analysis of route generation algorithm results	40
6.5.1 Advantages.....	40
6.5.2 Disadvantages	40
7 Discussion and conclusions	41
7.1 Evaluation of the techniques.....	41
7.2 Conclusions	42
7.3 Recommendations for further investigations.....	43
Appendices.....	45
References.....	46

1 Introduction

With the development of the economy, people's demand for tourism is growing. However, it is very difficult to plan a reasonable, economic and specific travel route in a short period. So, this project offers a convenient and fast way to plan the travel route for the user. The user only needs to input multiple destinations and the total travelling time and then the system recommends the best travel route based on the comprehensive consideration of the travel route cost. In addition, the system will also plan the detailed route between the travel spots in the city.

1.1 The subject matter and the scope of the investigation

Travel route planning is to develop a route in a certain area, to enable visitors to get the maximum viewing effect in the shortest time and have certain characteristics. And the route is reasonably connected by several tourist spots or tourist cities by the traffic line.

Under the constraints of attractions, transportation expenses, travel time, and travel locations, it is significant to investigate how to choose the right and best series of attractions from a variety of attractions and form a route for tourists to refer to. In terms of applications, it is necessary to combine attractions, itineraries, and routes and maps. When you select an attraction, you can see its location on the map. While planning the itinerary and route, you can also see the distribution of the itinerary and the direction of the route on the map.

1.2 The purpose of the project

It is very difficult to plan a reasonable, economic and specific travel route in a short period of time from plenty of travelling spots which can be shown in Figure 1-1. So, this system aims to provide a convenient and fast way to plan the travel route for the user.



Figure 1-1 Travelling Spots Diagram

Travelling Route Generation System offers a convenient and fast way to plan the travel route.

This web-based system contains at least two shining points. The first one is the system is clear and simple to use. The user only needs to input multiple destinations and the total travelling time and then the system recommends the best travel plan based on the comprehensive consideration of the whole travel route cost. The other one is the system

is fully responsive to phone and computers. Figure 1-2 shows the techniques used in the project.



Figure 1-2 Techniques used in the project

1.3 The organization of the report

This project report is divided into five chapters, each of which is as follows:

The first chapter introduces the background, purpose and challenge of the work of this project. And this report also illustrates related technologies, including travel itinerary planning algorithms.

The second chapter illustrates the analysis of this project.

The third chapter shows the techniques, algorithms and methodology used in the system.

The fourth chapter introduces whole Design and construction of this system.

The fifth chapter shows the algorithmic results and the presentation of the website.

The sixth part is about the analysis of the results above. And in the conclusions' section, critical evaluation of the techniques used, and

results obtained are carried out. And we also include recommendations for further investigations.

1.4 Survey of previously published work and current trends

The principle of determining the tourist route is based on the purpose of tourism, the physiological conditions of the tourist subject, the economic and time purchasing power of the tourists, the hobby of the tourists and the special tourism subject.

The traditional tourist attraction recommendation is typically based on the popularity of the attraction or the recommendation algorithm for personalized recommendation, but there are few studies considering the recommendation of the top-level hot spots. In terms of travel route planning, the algorithm for considering both time and cost is still a vacancy between two different points. This paper makes up for the lack of research on the recommendation of popular tourist attractions and multi-factor travel route planning. At the same time, the system integrates the functions from the scenic spots to the route to meet the needs of tourists.

As for how to generate the travelling route, the travel route planning problem is based on the evolution of the classic TSP problem. The TSP problem is a typical combinatorial optimization problem. At present, there are many researches on TSP in the world. It has been proposed based on the classical algorithm Dijk-stra algorithm [1], dynamic programming algorithm, branch and bound method [2], etc., and improved ant colony

based on heuristic optimization algorithm, genetic algorithm, simulated annealing algorithm, tabu search algorithm, Hopfieldl neural network, particle swarm optimization algorithm, immune algorithm, etc. In recent years, the research of Chinese domestic scholars mainly focuses on the application and optimization of ant colony algorithm in route planning.

Xu proposed the encounter algorithm based on the classical ant colony algorithm [6], which productively improved the quality of the ant colony algorithm. In addition, Xu improved the travel path, so that the ant colony algorithm can achieve dynamic planning.

In the article “Improving the Application of TSP Model in Optimal Tourism Route Planning”, he proposed an improved TSP model with multiple time constraints in combination with actual tour conditions (such as queuing). Xu has made reasonable adjustments and analysis in light of the various situations that may be encountered in real life.

Yang explicitly designed the constraints of tourism route planning in the article [7]. It includes the number of holidays and time requirements for current city residents, combined with time, distance and cost.

On the basis of Yang’s research, Wan improved the input variables of the whole system, including latitude and longitude coordinates, ticket fees, best travel routes and accommodation expenses, etc. in the article “Research on 5A Attractions Tourism Route Planning Problem Based on Ant Colony Algorithm”. Yang uses the ant colony algorithm to propose

four target benefit maximization models for comprehensive time, cost, distance and comfort which improves the accuracy of recommendation.

2 Analysis of problem

2.1 Data collection

In order to generate the inter-city play sequence, we need to find a suitable map API, enter the city name to get its latitude and longitude information, according to the mathematical formula, you can get the spherical distance between the cities, the city recommended the number of play days, the order of play between the attractions, the same reason, Given the name of the attraction or the address of the attraction, you can return to the route between the attractions and the time required.

Second, we also need to get some tourist information about the city, such as the recommended number of days in the city, for reference to the number of days in the city, the location of the city, the location of the attraction, and the recommended length of the attraction planning.

If we use crawlers to get the data, the speed of the crawler will greatly affect the speed of the program, so we need to design a file storage method, which can be read directly from the file for the input of the algorithm.

2.2 Simplification of the problem

In fact, the system of the project can be described in a more general way, which is to traverse these locations as much as possible within a fixed time given the distance between a series of locations and these locations. This is very similar to the TSP (Traveling Salesman Problem) problem.

The TSP problem is a typical combinatorial optimization problem. Many optimization problems can be directly or indirectly converted to TSP problems, and the TSP problem has been proved to have NPC computational complexity, which has great difficulty in solving.

TSP problem description:

Suppose there is a traveling businessman who wants to visit N cities, ask him to start from a city, visit each city at most once, and finally return to the starting city to ensure that the selected path length is the shortest.

On the basis of TSP, the system set more conditions. For example, the daily starting point and the ending point of the route within the city are hotels, and the daily playing time is within the time range corresponding to the compactness of the travel time selected by the user.

3 Methodology

3.1 Data collection

According to the above problem analysis, the first thing that needs to be done is the data collection part.

The city and attraction information are mainly crawled from commercial travelling websites. The system crawls multiple travel websites to get the required data. The route and distance information between cities and attractions comes from the map API. The API returns the required information such as the length and distance of the route and is used for system planning.

3.1.1 Web crawling

There is a wide variety of information on the website, and crawlers are often used when it is necessary to obtain some data with similarity and format. A crawler is an automated script that takes textual information about a web page and retrieves the information it needs based on a matching rule, which greatly reduces the labor of manually looking up data and viewing web pages.

The era of big data is coming, all applications are based on data-driven, and the first problem is to obtain these huge data. The traditional way may be manually search by a manual search engine. Although the search engine does help us to find the required data in countless web pages quickly, it also has many shortcomings: With the continuous advancement of the network, the form of resources has become more diverse from text, with pictures, videos, audio, and it is difficult for search engines to describe these files; under different application requirements, the data people care about may be

large. Not the same, and search engines often contain a lot of content that users may not care about, causing users to use the burden.

In order to solve the various shortcomings of the search engine, the system need to get a certain type of information quickly. That is why crawling is needed is this situation. The crawler can selectively access certain links and quickly obtain and integrate effective information.

3.1.2 Baidu Map API

The Baidu map front-end API is equipped with a JavaScript call interface, which can easily generate interactive pages such as front-end maps and route displays. It is very easy to develop on the browser or mobile phone, and this version also provides open source code for developers to download and view so that they can modify the open source library according to its own needs to simplify development.

Compared to Google map API, Baidu map API is more suitable for the use of mainland China. In comparison, Google map API can only get driving directions when generating routes between attractions, which is not suitable for the travel route recommendation system. In the use of the scene, more tourists will choose public transport such as bus and subway as a means of transportation between attractions. So when the system generates tourist routes, it considers the public transport method. The time, as well as the route plan that provides public transportation directly on the map, will be more in line with the user's needs.

3.2 Itinerary generation

3.2.1 Terminology definition

Total travel time: the duration of the travel the user needs to input.

Itinerary: the route generated by our website.

Travel spot: the hot spot we recommend to the user.

Play time: the estimated time to visit a certain travel spot.

Address of travel spot: the readable address of a travel spot.

Location of travel spot: the latitude and longitude representation of a travel spot.

3.2.2 Planning algorithm

In fact, the essence of the generation algorithm is to find a suitable shortest path among the candidate attractions. The starting and ending points of the path are all hotels, and the total length of the path is within the time range of the user-selected travel time.

There are many types of shortest path algorithms. The more famous algorithms include Dijkstra algorithm, Warshall-Floyd algorithm, dynamic programming algorithm, A* algorithm and improved Dijkstra algorithm. Even the same algorithm has many different implementations.

1. Dijkstra algorithm

Set the origin to be u_0 , and the target point is v_0 . The basic idea of the Dijkstra algorithm is to find the shortest path distance of each vertex of u_0 to G in order of distance u_0 from near to far, until v_0 , and the algorithm

ends. In order to avoid duplication and retain the calculation information of each step, a labeling algorithm is adopted.

The algorithm steps are as follows:

Step 1: Let $l(v_0) = 0$, for $v \neq u_0$, let $l(v) = \infty$, $S_0 = \{u_0\}$, $i = 0$;

Step 2: For each $v \in \overline{S_i} (\overline{S_i} = V \setminus S_i)$, use $\min\{l(v), l(u) + w(uv)\}$ instead of $l(v)$; When u, v are not adjacent, $w(uv) = \infty$. Calculate $\min\{l(v)\}$, and record a vertex that reaches this minimum as u_{i+1} , let $S_{i+1} = S_i \cup \{u_{i+1}\}$;

Step 3: If $i = |V| - 1$, stop; if $i < |V| - 1$, then $i + 1$ replaces i and go to step 2.

2. Warshall-Floyd algorithm

The Dijkstra algorithm is repeatedly executed N times with one vertex as the source point, but the Warshall-Floyd algorithm is more formally simple.

The basic idea of the Warshall-Floyd algorithm:

For each vertex $v_k \in V$, the shortest path from vertex v_i to vertex v_j passes or does not pass through the vertex v_k . Compare the values of d_{ij} and $d_{ik} + d_{kj}$. If $d_{ij} > d_{ik} + d_{kj}$, let $d_{ij} = d_{ik} + d_{kj}$, keep d_{ij} is the shortest path from the vertex v_i of the current search to the vertex v_j . This process is repeated, and finally, when all vertices v_k are searched, it is the shortest distance from vertex v_i to vertex v_j .

The basic steps of the Warshall-Floyd algorithm:

d_{ij} is the shortest distance from vertex v_i to vertex v_j , and w_{ij} is the weight from vertex v_i to vertex v_j .

Step 1: Enter the weight matrix W of the graph G , for all i, j ,

$$d_{ij} = w_{ij}, k = 1;$$

Step 2: Update d_{ij} , for all i, j , if $d_{ij} > d_{ik} + d_{kj}$, let $d_{ij} = d_{ik} + d_{kj}$;

Step 3: If $d_{ii} < 0$, then there is a negative loop with vertices v_i , stop;
if there is $k = n$, stop, otherwise go to step 2.

3. Dynamic programming algorithm

Dynamic programming algorithms are often used to solve the optimal solution problem. It is a tree branch problem, which means that a difficult big problem is decomposed into multiple small problems that can be easily solved, and then the answers obtained from the small problems are deformed and summarized. Thereby solving the big problem.

The algorithm steps are as follows:

Step 1: Find the nature of the optimal solution and characterize its structural features;

Step 2: Recursively define the optimal solution;

Step 3: Calculate the optimal solution in a bottom-up manner;

Step 4: Construct an optimal solution based on the information obtained when calculating the optimal value.

4.A* algorithm

The A* algorithm is based on the BFS algorithm (Breadth First Search Algorithm). It applies a valuation function to each node that is not expanded, compares the function values of these points, and selects the node with the highest estimate for expansion each time.

The A* algorithm is a graph search strategy in the field of artificial intelligence. Heuristic functions are used to evaluate the branches generated during the search process to select the best branch for searching. It more generally introduces a valuation function $f(n)$, which is defined as $f(n) = g(n) + h(n)$. Where $g(n)$ is the cost of reaching the current node, and $h(n)$ represents an estimate of the cost of reaching the target node from the current node, which must satisfy two conditions:

- (a) $h(n)$ must be less than or equal to the actual minimum cost of reaching the target node from the current node;
- (b) $f(n)$ must be kept monotonically increasing. [10]

The total number of nodes generated by the A* algorithm is N , and the depth of the solution is d . The consistent search tree attempting to be d must include the branching factor b^* in order to include $N + 1$ nodes. Therefore, $N + 1 = 1 + b_1 + b_2 + \dots + b_d$, it can be seen that the space complexity is high. If $h(n)$ is acceptable, that is, the heuristic function is well designed, the extended nodes can be greatly reduced, and the best path search can be completed in a small time. Its time complexity is $O(bd)$.

5. Improved Dijkstra algorithm

The traditional Dijkstra algorithm realizes the shortest path search between arbitrary nodes in the graph. The basic idea is to use the Dijkstra algorithm to calculate the shortest path starting from each node in the graph, so that any node in the graph can be obtained by looping n times. The shortest path between each, and each step is a simple iterative process. Although this can achieve the shortest path search between any two points, but the efficiency analysis is not optimal, in fact, it can be improved, the specific method is as follows:

Step 1: According to the Dijkstra algorithm, the speed of the shortest path between points can be improved by the access degree information of the nodes in the graph.

Step 2: After using the Dijkstra algorithm to find the shortest path between some nodes in the weighted graph, if there are other nodes that do not find the shortest path, system can use the shortest path information that has been found before to achieve fast shortest path lookup.

According to the analysis of the second subsection of the previous chapter, the system algorithm of this paper can be transformed into the TSP problem, but there are many different solutions to the TSP problem for different application scenarios.

3.2.3 Solutions for TSP problem

1. Greedy algorithm

Greedy algorithm is often used to solve optimization problems. At each step, it only considers the next optimal solution for the current one, and ignores the global optimality. This method is relatively optimal in the terms of locality, performance, efficiency. Because of these characteristics, the greedy algorithm has become the most widely used algorithm for solving TSP. However, for the global situation, the solution obtained by the greedy algorithm may be far from the global optimal solution, which is also the shortcoming of this algorithm.

2. Simulated annealing algorithm

Simulated Annealing (SA) algorithm is one of the effective methods to solve the TSP problem. It is easy to program and solve quickly. Simulated annealing is a global optimization algorithm. The stochastic state model is added to make the algorithm approximate the optimal state by probability selection. The convergence can be proved by rigorous theory. The simulated annealing algorithm has a complete set of annealing search plans, including a sufficiently high initial temperature, a slow annealing rate, numerous iterations, and sufficient probability perturbations [11].

3. Genetic algorithm

Genetic Algorithm (GA) Genetic Algorithm (GA) is inspired by Darwin's biological evolution theory and the evolutionary theory of biogenetics. It is a commonly used search optimal solution algorithm. The genetic algorithm was proposed by Professor J. Holland of the University of Michigan in 1975.

The algorithm usually consists of three parts: coding, individual fitness assessment and genetic operation. The genetic operations include chromosome replication, crossover, mutation and inversion [12].

4. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a new intelligent algorithm based on the mass model of bird foraging. It has strong versatility and parameters. Less, easy to implement, fast convergence, etc., is also one of the effective algorithms to solve the TSP problem.

5. Ant Colony Algorithm

Ant Colony Optimization (ACO) is a new bionic evolutionary algorithm in the field of optimization. The basic principle is derived from the shortest path principle of nature ants foraging, and the path that ants can walk in when searching for food sources. It releases an ant-specific secretion, pheromone, that allows other ants within a certain range to perceive and thereby influence their subsequent behavior. It was proposed by Marco Dorigo in his doctoral thesis in 1992. Ant colony algorithm is a kind of simulated evolutionary algorithm. It has many excellent properties including strong robustness. It is essentially a parallel distributed algorithm, which is easy to implement and can solve TSP problem well [13].

In our system, we will select as many places as possible that can be traversed according to the generated city play days and city hot spots list, but these selected spots may not be fully accessed due to the limitation of

the duration of the day trip. So the traditional TSP algorithm may not be suitable for our system, so we choose the most suitable and simple Greedy TSP algorithm from the above algorithms to generate route recommendations.

This paper mainly chooses the Dijkstra algorithm as the basis, using Greedy TSP algorithm as the framework, the steps are as follows:

- a) Start with a city and choose a city each time until all the cities are finished.
- b) Each time choose the next city, only consider the current situation, and ensure that the total distance of the path passed so far is the smallest.

3.2.4 Intra-city Route

The route planning between cities is mainly based on the geographical location of the starting city and each playing city. Taking into account the vehicle user when used to replace the tourist city there is a big difference, which directly led to the difference between the price and the time spent on the journey, such as: starting from Guangzhou, the destination is Beijing, train takes 30 hours, high-speed train takes 9 hours, and plane takes only 4 hours. The price between them is also different. It is difficult for the system to select a vehicle for the user and meet the needs of all users in time and money, so the specific itinerary recommendation between the cities is temporarily ignored, and the system only returns the recommended city play order and recommended play days as a result.

The specific algorithm is as follows:

The total play days W_D and the destination city name C_j are obtained from the user input, the C_j corresponding city information file is queried from the file storage system, the recommended travel days D_j of the city are obtained. According to W_D and $\sum D_j$, W_D is proportionally distributed,

$$R_j = \begin{cases} \text{round}(W_D \cdot D_j / \sum D_j), & \text{if } j \text{ is not the last city} \\ W_D - \sum_{i \neq j} R_i, & \text{else} \end{cases} \quad (3-1)$$

R_j represents the number of days in the final city j .

An array F is defined to indicate whether a city has been traversed.

When F_j is 1, it is traversed, and 0 is not traversed.

$$F_j = \begin{cases} 1, & \text{City } C_j \text{ has been traversed} \\ 0, & \text{City } C_j \text{ has not been traversed} \end{cases} \quad (3-2)$$

From the departure city C_{start} entered by the user, start traversing the city list C , select the city C_k closest to the current city as the next destination, and mark F_k as 1 and no longer traverse the city. Loop through this step until there is no city with a corresponding label of 0 in F , that is, all cities have been reached.

3.2.5 Inter-city Route

The route planning of specific attractions in the city is mainly divided into three parts: selecting candidate spots, recommending hotels, and generating daily routes.

The total length of daily play is derived from the compactness of the travel time selected by the user. The maximum value of the time limit is defined as T_{\max} and the minimum value is defined as T_{\min} .

First, according to the city play days R and the daily play time T generated in the foregoing steps, the longest duration $T_{\max} \cdot R$ that can be played for a plurality of days is obtained. Next, add the time of each attraction according to the order of the attractions. When the total length of the accumulated time $<$ the longest time to play $T_{\max} \cdot R$, continue to accumulate the next attraction, otherwise stop. Thus, the candidate attraction list S is obtained. The purpose of this step is to reduce the number of attractions and reduce the computational burden of the system in the subsequent route generation algorithm.

Since the starting point and the ending point of the daily tour are both hotels, in order to reduce the user's choice, the system provides the function of recommending the hotel. The hotel recommendation standard is the hotel comfort selected by the user on the form page, which respectively represents different hotel star standards. The system does this through the Baidu Map API.

Generating a daily route is the most important part of the Intra-city Route. Before starting, the attraction distance matrix M is generated, M is a square matrix of $m \cdot m$ dimensions, m represents the number of attractions in the candidate attraction list S , and M_{ij} represents the time

taken by the scenic spot i to the scenic spot j to take public transportation.

From Hotel H , traverse the list of attractions and find the attraction S_k that takes the least time from the hotel as the next attraction. Use S_k as a starting point to find the next attraction that takes the least amount of time. In line with the route planning between cities, create an array F of all the attractions, and record whether each attraction has been reached.

However, due to the limitation of the daily play time, the system adopts the following rules each time it decides whether to continue to go to the next attraction or return to the hotel:

When the current time $T_{curr} < T_{min}$ is used, try to traverse the next attraction, that is, select the attractions that take the least time in the unseen spots, but at the same time check the current used time T_{curr} + the cost of going to the next spot M_{ij} + suggested time for the next attraction P_j + next spot return to the hotel costs $M_{j,H}$ is greater than T_{max} . If it is greater than this, no longer go to the candidate spot today, and try to find other attractions as the next attraction with the same rules, if less than, the next location is the candidate attraction.

$$\text{Whether go to travel spot } k = \begin{cases} \text{no,} & T_{curr} + M_{ij} + P_j + M_{j,H} > T_{max} \\ \text{yes,} & \text{else} \end{cases} \quad (3-3)$$

After generating the one-day tour route, return the updated tag array and repeat the above steps to complete the multi-day route recommendation.

4 Design and construction of software system

This chapter will introduce the use cases, module interactions and the architecture of the system.

4.1 Use case modeling

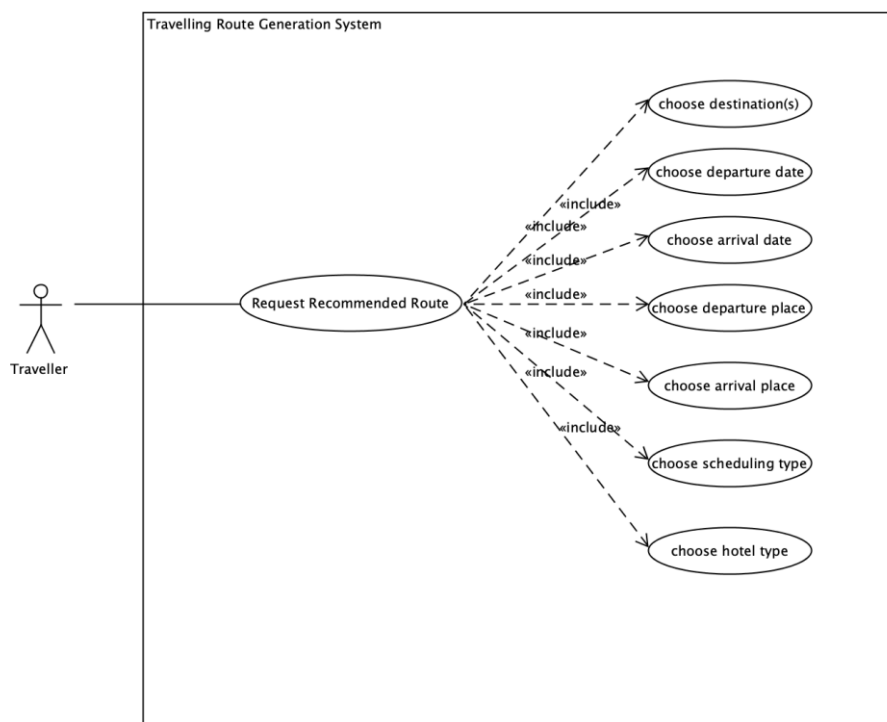


Figure 4-1 Use Case Diagram

Main success scenario is that a user first visit to our website and input the desired destination(s), departure date, arrival date, departure place, arrival place, scheduling type and the hotel type. Then the user can submit the form to the system. The system will return a route to the user. Figure 4-1 shows the use case diagram of the Travelling Route Generation System.

Alternative scenarios are that if any of the above fields are empty, or that the departure date or the arrival date is not date type, or that the arrival date is not before the departure date, the user needs to input the correct field again. Other failed scenarios may contain but not limit to the missing cities or other fields in data and error returned from API and crawler.

4.2 Module Interactions

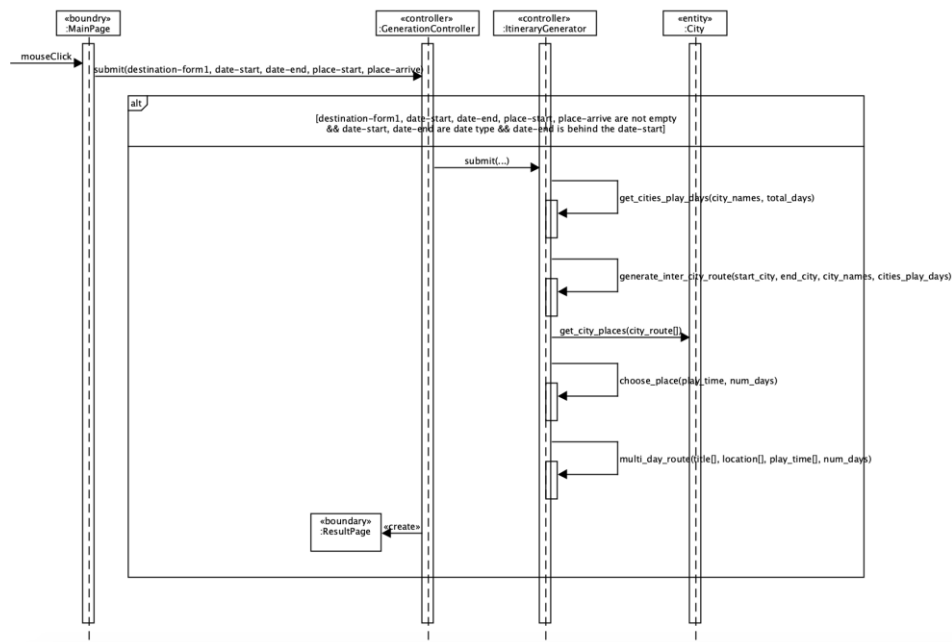


Figure 4-2 Sequence Diagram

Figure 4-2 shows a system view of the main success scenario. The user will input the destination(s), departure date, arrival date, departure place, arrival place, scheduling type and the hotel type and click the submit button. The GenerationController will handle the request and call the ItineraryGenerator to generate the route. Data crawled from the ctrip and qunar is fetched by the ItineraryGenerator to generate the route. Then the

route will be returned to the GenerationController. Finally, the GenerationController will parse the route and return the result page to the user.

4.3 Architecture Modeling

Figure 4-3 shows the architecture of Travelling Route Generation System and the specific artifacts of the system. Our system mainly contains two subsystems.

The Web Crawler subsystem first crawls the city name and corresponding city code from ctrip.com website and saves them in the city_list.csv file. With the help of the previous crawled city code, the Web Crawler subsystem then crawls the city data from ctrip.com and save into the [citycode].json file. The city data includes the city name, recommended play time and the travel spots. Travel spots are saved in such an order that the first one is the most popular travel spot.

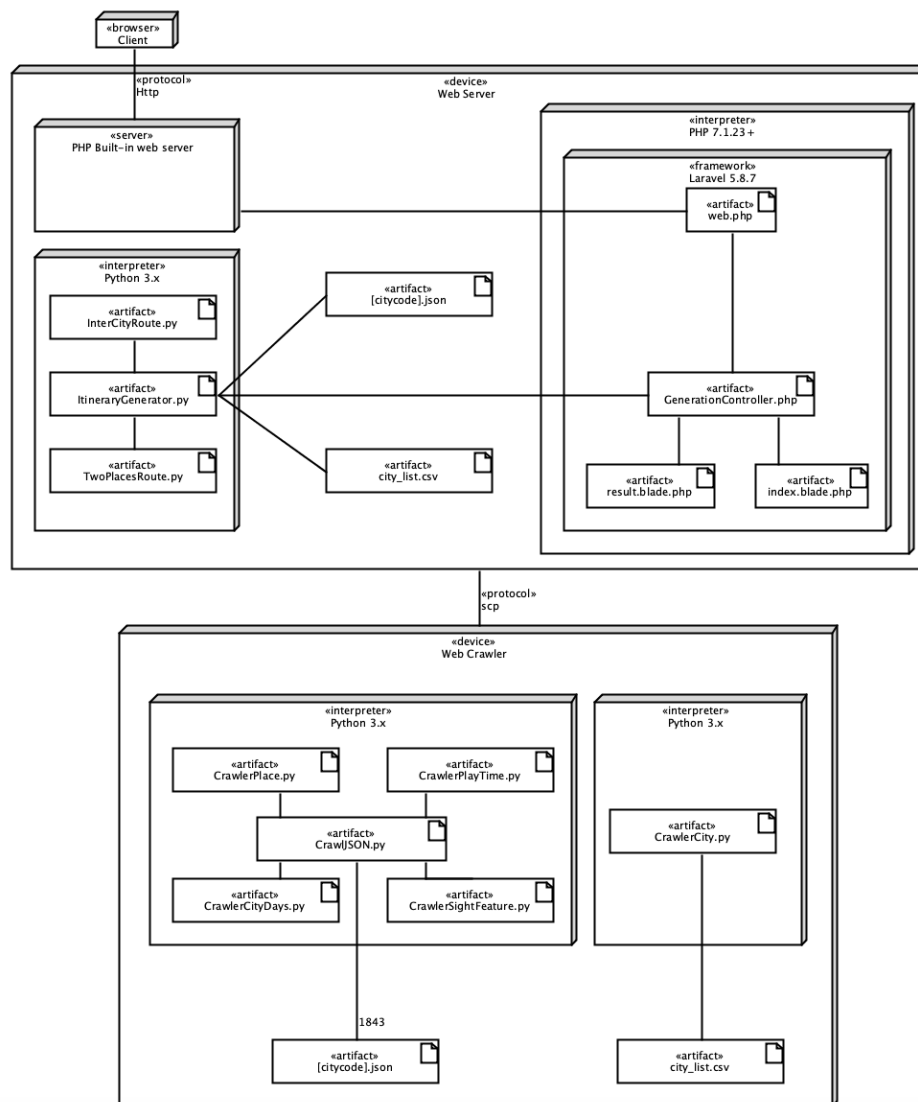


Figure 4-3 Deployment Diagram

The Web Server subsystem is responsible for the generation of the route. At the very beginning, the user sends an HTTP get request to our server. The web.php will redirect the request to the index function of the GenerationController.php to return the main page to the user. After the user submits a form, an HTTP post request will be sent to our server. The web.php will call the generate function of the GenerationController.php to generate the route. In the generate function, the fields of the submitted form are obtained and passed as the arguments to the

ItineraryGenerator.py. InterCityRoute.py and TwoPlacesRoute.py as well as the city_list.csv and [citycode.json] is used to assist to generate the route. After the route is created, GenerationController.php obtains the route and passes it to a JavaScript function. In the result page, JavaScript and JQuery is used to transform the route from the ItineraryGenerator.py into some arguments and pass to corresponding interface so that the user can get the final result page.

In the end, the Web Server subsystem can serve the user while the Web Crawler subsystem can update the data periodically in the back-end. This design is better than the original one which only saves the city_list.json file and crawls the city and travel spot information only when that information is requested by the user.

5 Algorithmic and experimental results

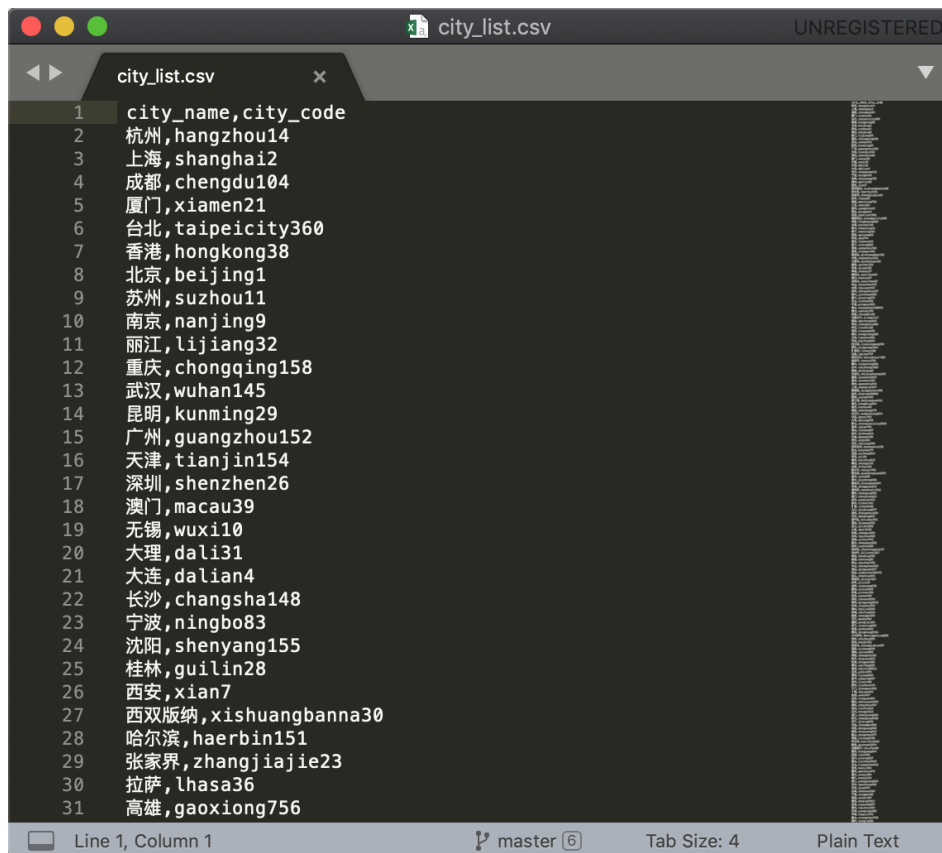
This chapter mainly discuss the artifacts of the chapter four.

5.1 Web Crawler subsystem

According to the chapter four, city_list.csv is the start point of our system. Figure 5-1 display the content of the file which includes the city name and the corresponding city code. The city_list.csv file is obtained from the ctrip because the most fields of [citycode].json are obtained from ctrip too.

[citycode].json files are the next artifacts to be created in our system. Figure 5-2 display the content of the shenzhen26.json file. The cityName

field represent the name of Shenzhen. The days field represent the recommended play day of Shenzhen. The cityImage field is empty now but may be used for the future version of our system. The spots field stores an array of travel spots in Shenzhen. Each travel spot contains the title, address, image, rank, time, location and feature fields. The title field represents the name of a travel spot. The address field represents the address of a travel spot. The image field is empty now but may be used for the future version of our system. The rank field represents the ranking of a travel spot among all the travel spots in Shenzhen. The time field represents the recommend play time in a travel spot. The location stores the longitude and latitude of a travel spot. The feature fields represent the description of a travel spot.



```
1 city_name,city_code
2 杭州,hangzhou14
3 上海,shanghai2
4 成都,chengdu104
5 厦门,xiamen21
6 台北,taipeicity360
7 香港,hongkong38
8 北京,beijing1
9 苏州,suzhou11
10 南京,nanjing9
11 丽江,lijiang32
12 重庆,chongqing158
13 武汉,wuhan145
14 昆明,kunming29
15 广州,guangzhou152
16 天津,tianjin154
17 深圳,shenzhen26
18 澳门,macau39
19 无锡,wuxi10
20 大理,dali31
21 大连,dalian4
22 长沙,changsha148
23 宁波,ningbo83
24 沈阳,shenyang155
25 桂林,guilin28
26 西安,xian7
27 西双版纳,xishuangbanna30
28 哈尔滨,haerbin151
29 张家界,zhangjiajie23
30 拉萨,lhasa36
31 高雄,gaoxiong756
```

Figure 5-1 City_list.csv File


```

CrawlerJSON call stack
CrawlJSON:main():
    CrawlJSON:crawl_all_city(startline, endline)
        CrawlJSON:get_index_url(item[1], num_pages + 1)
            CrawlJSON:get_pages_from(city_code, num_pages)
CrawlJSON:get_html(index_url)
CrawlJSON:parse_html(html, index)
    CrawlJSON:get_lng_lat(address)
    CrawlerSightFeature:get_sight_feature(sight_url)
    CrawlerPlayTime:get_play_time(title)
    CrawlerCityDays:get_city_play_days(item[0])
    CrawlerCityDays:get_html(search_url)
    CrawlerCityDays:parse_city_html(html)
    CrawlerCityDays:get_html(city_url)
    CrawlerCityDays:parse_detail_html(city_html)

```

Figure 5-4 CrawlJSON.py Call Stack

Figure 5-4 displayed the call stack to crawl the [citycode].json file. This figure shows the relationship among CrawlJSON.py, CrawlerCityDays.py, CrawlerPlayTime.py, CrawlerSightFeature.py. The absence of the CrawlerPlace.py file in the call stack is because the functions of the CrawlerPlace.py are merged into the CrawlJSON.py but we still consider the CrawlerPlace.py a part of the Web Crawler subsystem.

5.2 Web Server subsystem

After crawling all the files, the generation of route becomes possible. With the help of the Baidu API, the hotel recommendation is implemented in the main function of the ItineraryGenerator.py. Our system can provide three hotel types which are luxury, normal and economics. The luxury will recommend the most expensive hotel, the economics will recommend the cheapest hotel and the normal will recommend the most popular hotel for the user. Additionally, our system can also provide three scheduling types which are loose, moderate and compact. These parameters will affect the

play time of each day. Specifically, the user can spend at most eleven hours on travel with the loose scheduling type. The number is twelve with the moderate type and thirteen with the compact type.

```

ItineraryGenerator call stack
ItineraryGenerator:main()
  CrawlerCityDays:get_cities_play_days(city_names, total_days)
    CrawlerCityDays:get_city_play_days(city_name)
      CrawlerCityDays:read_csv(city_name)
        CrawlerCityDays:allocate_time(cities_play_days, total_days, count_days)
      InterCityRoute:generate_inter_city_route(start_city, end_city, city_names, cities_play_days)
        InterCityRoute:inter_city_transit(get_lng_lat(curr_city), get_lng_lat(city_names[j]))
        CrawlerPlace:get_lng_lat(curr_city)
      CrawlerPlace:get_city_places(city_route[i])
        CrawlerPlace:read_csv(city_name)
      ItineraryGenerator:choose_place(play_time, num_days)
      ItineraryGenerator:multi_day_route(hotelName, hotelLocation, title, location, play_time, num_days)
        TwoPlacesRoute:transit(origin, destination)
        ItineraryGenerator:one_day_route(hotelName, hotelLocation, title, location, play_time,
        place_flag, hotel_time_matrix, hotel_return_time_matrix, time_matrix)

```

Figure 5-5 ItineraryGenerator.py Call Stack

```

ubuntu@ip-172-31-16-11:~$ python3 TravelPlace/ItineraryGenerator.py 香港 北京 7 Luxury Loose 三亚 深圳 上海
香港|北京
深圳|三亚|上海
2|3|2
深圳福田香格里拉大酒店|世界之窗|深圳欢乐谷|锦绣中华民俗村&0|3.5|2.5|4.0&114.063501|113.980244|10292683|113.9
8894057790503|113.98894057790503&22.542043|22.542195480333724|22.54079698453105|22.54079698453105
深圳福田香格里拉大酒店|红树林|深圳野生动物园|杨梅坑&0|2.5|2.0|2.5&114.063501|114.0699869119043|113.96719141
93456|114.25445487896184&22.542043|22.534439477374193|22.597301948641412|22.72601655078104
三亚美高梅度假酒店|亚龙湾热带天堂森林公园|亚龙湾|亚龙湾海底世界&0|4.0|4.0|3.0&109.642533|109.65013564160743
|109.65013564160743|109.66313419150204&18.234049|18.23821057515121|18.23821057515121|18.23906124490067
三亚美高梅度假酒店|三亚千古情景区|春园海鲜广场|第一市场|椰梦长廊&0|1.5|2|2|1.5&109.642533|109.5370681917320
6|109.51078100034796|109.5148714156219|109.48634330412692&18.234049|18.296081291679297|18.26535302195487|18
.247042117051858|18.277528704653676
三亚美高梅度假酒店|大东海|鹿回头风景区&0|3.6|3.0&109.642533|109.5277753541836|109.50810136399565&18.234049|
18.226394144542454|18.231760426927504
上海外滩W酒店|外滩|南京路步行街|上海海洋水族馆|东方明珠|上海环球金融中心&0|2.0|2.5|2.0|2.5|2.0&121.502959|1
21.49687990965766|121.49224712446464|121.50854982169035|121.5063729732192|121.51420808370779&31.254676|31.2
42706236913193|31.24108819373416|31.246011505504946|31.244937070519477|31.240286930090868
上海外滩W酒店|陆家嘴|田子坊|城隍庙旅游区&0|2.5|3.5|2.5&121.502959|121.55045460683195|121.47602394434436|121
.49918988251385&31.254676|31.227348292436346|31.214225106535267|31.231085650198086
ubuntu@ip-172-31-16-11:~$

```

Figure 5-6 Results of ItineraryGenerator.py

Figure 5-5 is the call stack of the ItineraryGenerator.py, which shows the relationship with the InterCityRoute.py and TwoPlacesRoute.py. The detailed algorithm has been given in the previous chapters. Figure 5-6 is an example of the output of the ItineraryGenerator.py. This result may seem to be a little obscure, but it can be easily handled later by the JavaScript program.

```

$arguments = $request->get('place-start') . " " . $request->get('place-arrive') . " " . $total_day . " " . $hotelLuxury . " " .
$$scheduledType . " " . $request->get('destination-form1') . " " . $request->get('destination-form2') . " " . $request->get('destination-form3');
$commandstr = "python3 /Users/laulingai/Desktop/Travelling-Route-Generation-System/TravelPlace/ItineraryGenerator.py " . $arguments;
$command = escapeshellcmd($commandstr);
$output = shell_exec($command);
return view('routes.result', compact('output'));

```

Figure 5-7 GenerationController.php Call ItineraryGenerator.py

The following parts are mainly about the front-end design. Since our system uses the Laravel framework which is a classical MVC framework, to get the user input, a view called index.blade.php is created. The user inputs the fields and pass the form to the controller GenerationController.php. GenerationController.php extracts the corresponding information, calculate the total play day by the departure date and arrival date, converts the inputs into the desired format as in Figure 5-6 and calls the ItineraryGenerator.py script as Figure 5-7. Figure 5-7 also reveals that immediately after the ItineraryGenerator.py returns the route. The result is passed a view called result.blade.php. The remaining works will be handled by the JavaScript of the result.blade.php. This design is based on the desire to alleviate the server pressure.

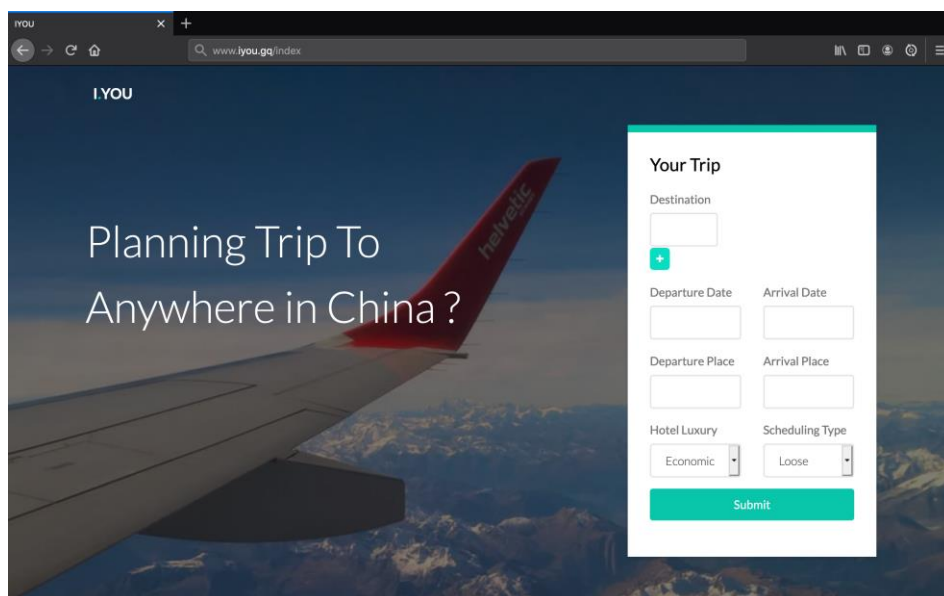


Figure 5-8 Travelling Route Generating System Main Page

The development of the front end took a lot of time and effort because our system aims at providing user-friendly interfaces for the user. The

detailed codes can be fetched from the GitHub link in the appendices. In this report we will only focus on the results of the website.

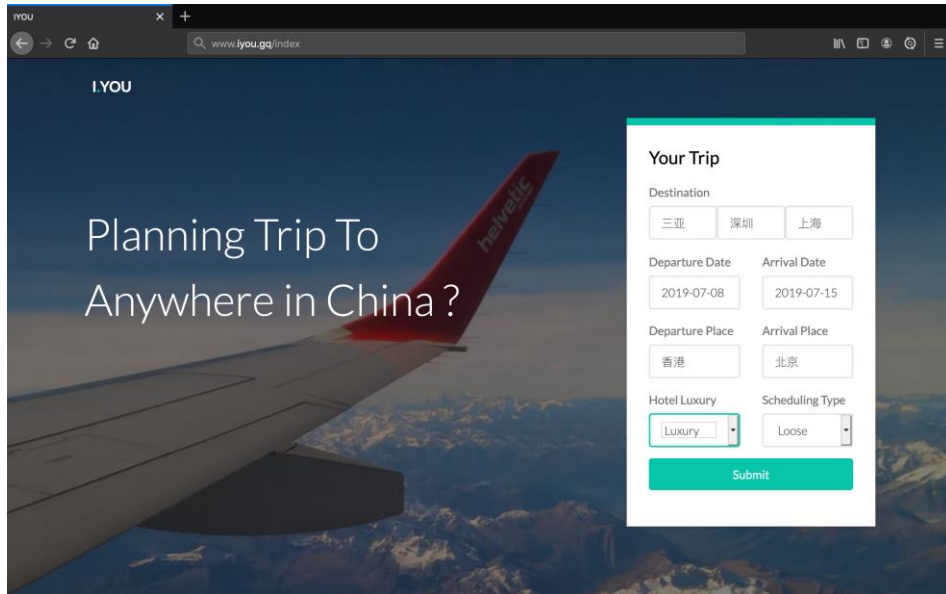


Figure 5-9 After Clicking the Add Button and Input the Fields on Main Page

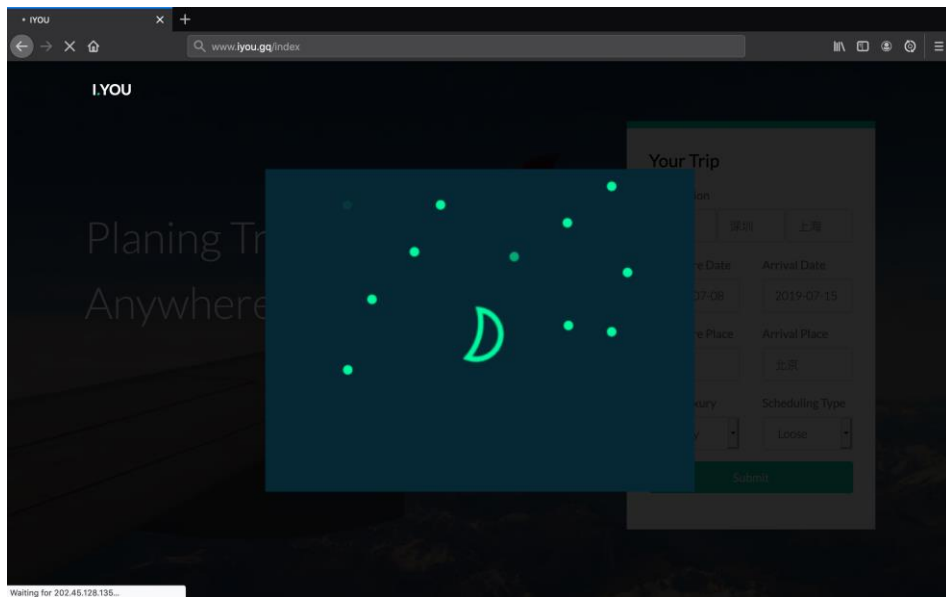


Figure 5-10 Loading Image After Clicking the Submit Button

Figure 5-8 displays the main page of the website. The user can input at most three destinations. Figure 5-9 is an example of input. Figure 5-10 shows that the loading image after clicking the submit button.

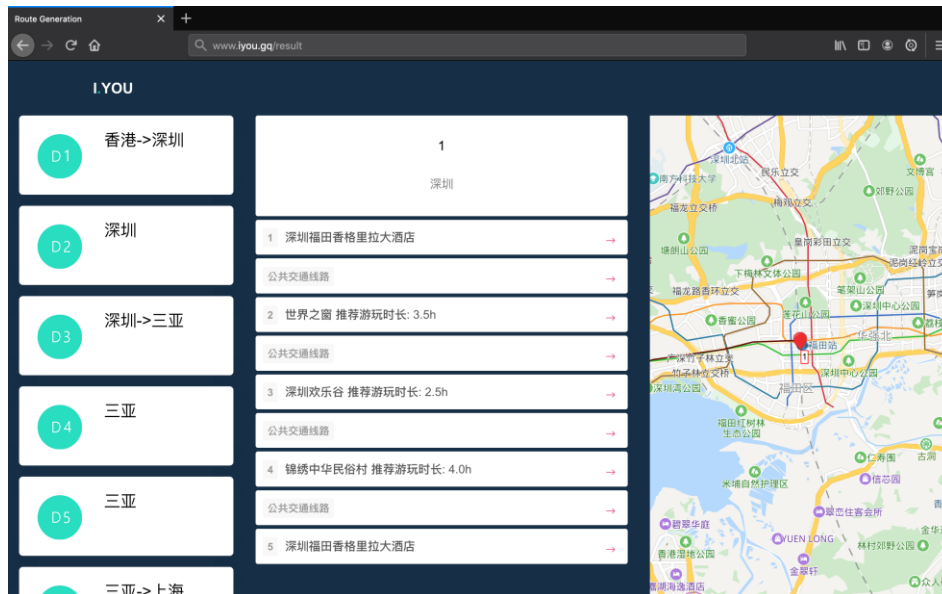


Figure 5-11 Initial Result Page



Figure 5-12 Selecting a Specific Day and Choosing a Travel Spot

If the user can reach the result page similar to figure 5-11, it means the system has successfully generated a route for them. The D1 in the green circle stands for day one. The user can click the green circle to obtain the corresponding route of a specific day. Plus, the I.YOU logo on

the top left of the result page is attached an event listener which can go back to the main page to submit the form again.

As shown in the Figure 5-12, the spots name and recommended play time are displayed in the middle part of the result page while the Baidu Map are displayed on the right-hand side. The user can click a spot to obtain an intuitive view of the location on the map. There is a number label near the red mark of the location of each travel spot.

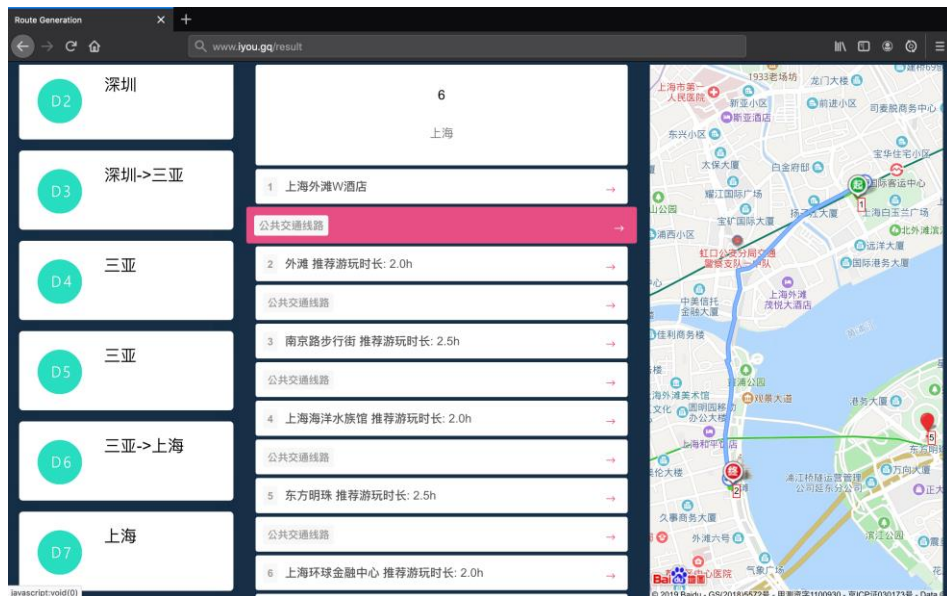


Figure 5-13 Transportation Route on the Baidu Map

As shown in the Figure 5-13, the user can click a transportation route between spots to obtain an intuitive view on the map. The “起” in green is the start point and the “终” in red is the terminal point in a transportation route.



Figure 5-14 Detailed Information about the Transportation Route

As shown in the Figure 5-14, the user can click the “起” on the map and get the detailed location to get on and exit the bus until the user reaches travel spots. The bus number is also indicated as a part of information.

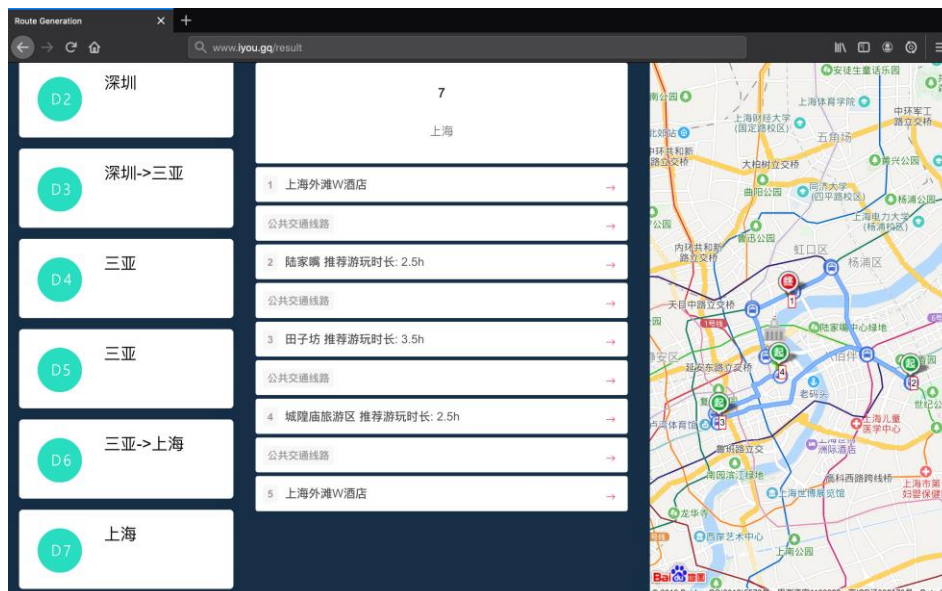


Figure 5-15 All the Transportation Route Displayed on Baidu Map

As shown in the figure 5-15, the total play day of the route is seven.

In a particular date, the route will start from the hotel and back to the hotel. If the user clicks many transportation route, Baidu Map can display multiple routes as well. The transportation route will disappear after the user switch to another day. The user needs to click the transportation route to regain the view of the transportation route on the map.

6 Analysis of results

6.1 Analysis of data collection results

This project uses Ctrip.com as the information source of city names, attraction names, attraction recommendation rankings and addresses.

In addition, the use of Qunar.com as the information source of the attraction time and the city information time.

The Baidu Maps API serves as a source of information for obtaining latitude and longitude data of scenic spots and cities, and routes between scenic spots.

6.1.1 Advantages

- a) Using the travel website as a data source, it is very straightforward and convenient to get the desired DIV from the HTML code of the web page.
- b) Using the Baidu Maps API, we can quickly and accurately get the latitude and longitude corresponding to the address and the transit routes between the attractions.

- c) The reason for choosing different travel websites as data sources is that the information of a single travelling website is incomplete and inaccurate, and multiple travel websites as information sources can ensure the accuracy of the data.

6.1.2 Disadvantages

- a) Data sources cannot guarantee reasonable travelling time data. Because it is relatively subjective tourism data, it changes with people's preferences.
- b) Multiple travel sites cannot guarantee the matching of special data. For example, a certain attraction can be found on the Ctrip.com, but it cannot be found on the Qunar.com.

6.2 Analysis of web crawling results

This project gets the html code directly through the page url and then parses the html to get the required elements.

6.2.1 Advantages

- a) Separate crawler data will make the data more transparent. Therefore, developers will also feel a sense of security.
- b) The data of the crawler can be directly called by the route planning algorithm, which makes the performance of the algorithm greatly improved.

6.2.2 Disadvantages

- a) Parsing html can only grab some specific data. If the page uses Ajax asynchronous requests, we can't crawl this part of the data.

- b) The dependency on the page's HTML code is too large, and once the site is updated, the crawler code will fail. This requires routine maintenance and its frequent work.

6.3 Analysis of map API results

6.3.1 Advantages

- a) From the perspective of most O2O developers, the role of maps can be more understood as the need for "positioning", which is to meet the "peripheral" option in the product. Baidu map API is accurate and the surrounding resources are relatively rich. This is a good fit for the needs of this project.
- b) Baidu Maps API products are faster to iterate. It works well in real-world scenarios.
- c) Compared to Google Maps API, Baidu Maps API can offer stable and accurate transit route in mainland China without high usage costs which cannot be offered by using Google Maps API.

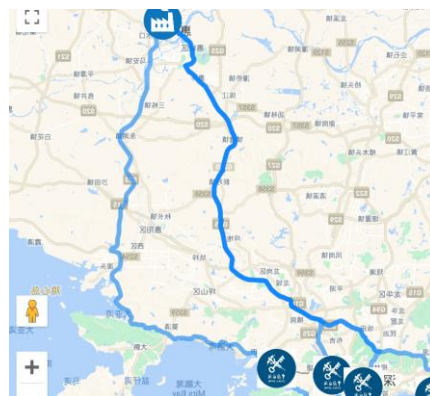


Figure 6-1 Google map API



Figure 6-2 Baidu map API

6.3.2 Disadvantages

- a) Compared to the Google Maps API and the Apple Maps API, Baidu's map data system is relatively independent. It caused the data of Google Maps not to be accurately positioned on Baidu Maps.
- b) The source of information about the attractions of Baidu Map is unknown and unsafe. This can easily lead to bad users' experience and even users' complaints.
- c) Compared to the Google Maps API, the Baidu Map API ensures the normal display of transit routes. But it has lost some of the display features, such as the ability to display multiple routes at the same time.
- d) Due to the limited navigation function in Mainland China, Baidu Maps API cannot support other countries except China. This will be solved by changing API abroad.

6.4 Analysis of website development results

This project mainly uses PHP's Laravel framework. This enables the entire project to be managed by the team, and it speeds up development.

6.4.1 Advantages

- a) The framework can better organize code and files.
- b) It can provide a huge amount of public code and class libraries. For example, it provides code for form validation, input and output data filtering, database abstraction, session and cookie handling, email, calendar, and paging. This greatly speeds up development.
- c) For a framework, most of the filtering work can be done automatically. For example, any value passed through the database will be filtered to prevent sql injection attacks. All html can be generated automatically. It is easy to change a configuration option to encrypt a cookie.

6.4.2 Disadvantages

- a) The Laravel framework is a component-based framework. Therefore, it is bloated and reduces the performance of PHP.
- b) There are fewer examples of official Laravel framework documentation. It has less guidance and consideration for the Model layer. This has led to a significant increase in the time we spend on learning the framework.

6.5 Analysis of route generation algorithm results

The algorithm is based on the greedy algorithm of the TSP problem. The algorithm can realize the route planning of many cities as well as multiple attractions.

6.5.1 Advantages

- a) It takes the length of the traffic between the two attractions instead of the distance as the cost. This improves the utility and accuracy of the algorithm for travelers.
- b) In addition, it incorporates the recommended score into the criteria for measuring whether to go to the next attraction.

6.5.2 Disadvantages

- a) The time complexity of the algorithm is $O(n^2)$. Performance is not greatly improved on the basis of greedy algorithms.
- b) Every time you get the traffic duration between two points, the algorithm needs to call the Baidu Map API, which slows down its speed.
- c) The algorithm does not balance the performance of the algorithm with the freedom of the user's choice of attractions well. If the algorithm improves performance by lowering n , the number of spots the user can select is bound to be reduced, which makes the travel route not change greatly, and thus the users' experience will be decreased.

7 Discussion and conclusions

7.1 Evaluation of the techniques

From the test results of the website (Figure 7-1), it can provide a route with high popularity and reasonable planning between cities.



Figure 7-1 Results of Nanjing from project

Figure 7-1 shows that algorithm generates a reasonable route between countries. And we can also conclude that it generates a complete and interesting route every day.



Figure 7-2 Results of Nanjing from another similar website

Compared with the results from another similar website, the route of this project is better and abundant. In addition, other similar website cannot offer transit way between spots but only driving mode which is apparently not suitable for economic travelers.

7.2 Conclusions

From the perspective of website development, this website uses Laravel framework as the server framework, PHP language to write background code, JavaScript language to design front-end effects and Baidu Maps API to better present the route intuitively. Based on the implementation of the test route, it adds an interface that uses maps to display the route visually. And the website also provides an interface that can be used for navigation in real life.

From the perspective of planning route algorithm, this project proposes a dynamic travel route planning algorithm based on optimal route planning, which comprehensively considers the shortest travel time and recommended ranking. By dynamic recursion, the route with the smallest iteration of planning motivation is the optimal tourist route. The practical application proves that the algorithm can output the best route and it meets the general rules of the travel and ferry process. The results meet the individual needs of the tourists so as to satisfy the tourists' best motivation benefits.

7.3 Recommendations for further investigations

This website runs slowly on Amazon servers (It takes about 2 minutes to show a city's travel itinerary). If it is built on the GPU provided by the CS department, it will take about 30 seconds. But this is still very slow for a functional website. There are two main reasons for the slowdown of the website. One is the limitation of the algorithm itself. We have not been able to design an algorithm with time complexity less than $O(n^2)$. The other is due to the frequent invocation of the API, when judging whether the attraction is appropriate.

Based on the above analysis, the project will focus on the design of better performance algorithms in the future. One direction is to increase the speed by lowering N , but this also reduces the number of selectable attractions, thereby reducing the user experience. Another direction is to apply the idea of branch and bound to the current algorithm. In the worst case, the time complexity of Branch and Bound remains same as that of the Brute Force clearly in worst case where we may never get a chance to prune a node. Whereas, in practice it performs very well depending on the different instance of the TSP. The complexity also depends on the choice of the bounding function as they are the ones deciding how many nodes to be pruned.

In addition, this website has not yet implemented the display of traffic information between cities on the website. In the future, we will focus on

adding inter-city transportation information (including airplanes, trains, ships, etc.) to achieve continuity of the website process and enhance the user experience.

Appendices

Github link: <https://github.com/pbs-lzy/Travelling-Route-Generation-System>

References

- [1] Dijkstra, Edsger W, Chapter I: Notes on structured programming
Structured programming, 1972.
- [2] Lawler E L, Wood, D. E, “Branch-and-Bound Methods: A Survey”.
Operations Research, 1966, 14(4):699-719.
- [3] Yang J, Shi X, Marchese M, et al, “An ant colony optimization method
for generalized TSP problem”, Progress in Natural Science, 2008.
- [4] Fuchs H, Kedem Z M, Uselton S P, “Optimal surface reconstruction
from planar contours”, Communications of the Acm, 1977.
- [5] Sakoe H, Chiba S, “Dynamic programming algorithm optimization for
spoken word recognition”, IEEE Transactions on Acoustics, Speech, and
Signal Processing, 1978.
- [6] Xu Feng, “Improving the Application of TSP Model in Optimal Tourism
Route Planning”, journal6, 2006.
- [7] Yang Yunpeng, et al, “Research on 5A Attractions Tourism Route
Planning Problem Based on Ant Colony Algorithm”, Practice and
understanding of mathematics, 2016.
- [8] Norkin V I, Pflug G C, Ruszczyński A, “A Branch and Bound Method
for Stochastic Global Optimization”. Mathematical Programming, 1998.
- [9] Silva R, Lopes H S, Godoy W, “A Heuristic Algorithm Based on Ant
Colony Optimization for Multi-objective Routing in Vehicle Ad Hoc
Networks”, 2013 BRICS Congress on Computational Intelligence & 11th

Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC).

IEEE, 2013.

[10] Bin Yang, Xinzhu Sang, Shujun Xing, Huilong Cui, Binbin Yan, Chongxiu Yu, Wenhua Dou, Liquan Xiao, *A-star algorithm based path planning for the glasses-free three-dimensional display system*, SPIE/COS Photonics Asia, 2016.

[11] Anzhi Qi, *Research on TSP Solution Based on Improved Simulated Annealing Algorithm*, Proceedings of 2017 4th International Conference on Machinery, Materials and Computer (MACMC 2017), 2017:4.

[12] Gambardella, L.M., Dorigo, M., *Solving symmetric and asymmetric TSPs by ant colonies*, 1996,.

[13] Lijun Sun, *Genetic Algorithm for TSP Problem*, Proceedings of 2015 International Industrial Informatics and Computer Engineering Conference (IIIEEC 2015), 2015:4.