A constraint-based route search system for smart phone in attraction facilities

Takahiro Shibuya
Faculty of Science and Technology,
Tokyo University of Science,
2641 Yamazaki, Noda,Chiba, 278-8510, Japan
j7411616@ed.noda.tus.ac.jp

and

Hayato Ohwada
Faculty of Science and Technology,
Tokyo University of Science,
2641 Yamazaki, Noda,Chiba, 278-8510, Japan
ohwada@ia.noda.tus.ac.jp

ABSTRACT

It is very difficult for us to find the minimum route to travel in attraction facilities. A searching system for visitors would be useful. Therefore, we constructed a system to find route with the minimum total traveling time. Facility visitors can employ this system on a smart phone. The system is composed of ECLiPSe (ECLiPSe is not a software development environment) and Java Servlet. We concluded that our system is useful and can greatly shorten travel time within the facility.

Keywords: traveling salesman problem, traveling problem in attraction facilities, smart phone, ECLiPSE, Java Servlet,

1. INTRODUCTION

There are many attractions in popular attraction facilities, making it difficult for visitors to find the fastest way of moving about. A visitor picking a very slow route may become tired from walking and waiting, and may miss the opportunity to ride the desired attraction.

I think that visitors can move around quickly if there is a system to find the fastest order.

This traveling problem is similar to the traveling salesman problem. A traveling salesman must find the shortest possible route that visits each city exactly once, when given a list of cities and their pairwise distances.

Research is being conducted to solve the attraction routing problem by applying the traveling salesman problem. For example, research is being conducted to propose how best to move around the "2005 World Exposition, in Aichi, Japan." with two-opt method and a simulated annealing method, which is a one of meta-heuristics method to search for an approximate solution cite.[1]

Research is also being conducted to propose how to travel efficiently with CPLEX.[2]

However, such research has two problems. First, the research employs a fixed waiting time and therefore is not a realistic model. Second, we cannot actually use these systems.

Our goal is thus to develop a realistic model and a route search system that visitors can use on a smart phone.

2. SUBJECT ATTRACTIONS

Subject area

We construct a system for Tokyo Disneyland in Chiba Prefecture in Japan as an example. A visitor chooses eight of the thirty-one attractions he/she would like to visit. We assume that the visitor can ride all attractions without considering service being suspended.

Location of Attraction

Figure 1 presents a map of Tokyo Disneyland, and Table 1 lists the attraction's number, name, and argument. The attraction's argument means the attraction's name in the system. The shortest distance between attractions is measured with "Kyorisoku"[10], a map service to measure distance provided by Mapion Co.,Ltd.(a Japanese company). When we measure distances, we use things to be measured on the map with Kyorisoku. We regard walking speed as three kilometers per hour and set it for the transit time.

We assume that the attraction's entrance and exit are at the center of the attraction and that the center point leads to the nearest street because we cannot determine the attraction's entrance and exit from the map.

We adopt the data of official site in Tokyo Disneyland[11] as the seat-load time. Actual waiting times were published in October 2008 by Tokyo Disneyland. We adopt holiday's and weekday's every thirty minutes waiting times. We utilized the site "Congestion expectation calendar in Tokyo Disneyland"[11] to adopt waiting time.

Business hours actually differ by date, but we assumed that Tokyo Disneyland is open from 8 a.m. to 10 p.m.

Table 1
Attraction in Tokuyo Disneyland

number	name of attraction	argument
1	Splash Mountain	splash
2	Space Mountain	space
3	Pirates of the Caribbean	carib
4	The Haunted Mansion	haunted
5	Big Thunder Mountain	big
6	Pooh's Hunny Hunt	hunny
7	It's a Small World	smallworld
8	Star Tours	startours
9	Jungle Cruise	jungle
10	Western River Railroad	westernriver
11	The Enchanted Tiki Room	tiki
12	Westernland Shootin' Gallery	westernlandshoot
13	Country Bear Theater	bear
14	Mark Twain Rivarboat	marktwain
15	Tom Sawyer Island Rafts	ikada
16	Beaver Brothers Explorer Canoes	canoes
17	Peter Pan's Flight	peterpan
18	Snow White's Scary Adventures	shirayuki
19	Pinocchio's Daring Journey	pinocchio
20	Dumbo The Flying Elephant	dumbo
21	King Arthur Carrousel	carrousel
22	Alice's Tea Party	alice
23	Roger Rabbit's Car Toon Spin	cartoon
24	Minnie's House	minniehouse
25	Mickey's House and Meet Mickey	mickeyhouse
26	Gadget's Go Coaster	gocoaster
27	Chip'n Dale's Treehouse	treehous
28	Donald's Boat	donaldboat
29	Buzz Lightyear	buzz
30	Star Jet	starjet
31	Autopia	raceway

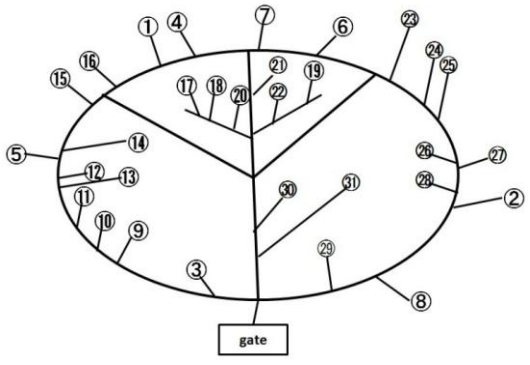

Figure 1. Attraction Map

3. SYSTEM OUTLINE

ECLiPSe Program

We constructed a system with ECLiPSE. Though it often makes mistakes, the ECLiPSe we are using is not a multi-language software development environment comprising an integrated development environment (IDE). ECLiPSe is a software system for developing and deploying Constraint Programming applications. For example, it is useful in the areas of optimization, planning, scheduling, resource allocation, timetabling, transport, etc. The ECLiPSe language is largely backward-compatible with Prolog and supports different dialects.

This system has one program to calculate the minimum time when considering transit time, waiting time, and seat-load time. A visitor inputs the starting time, the attraction's name(from Table 1), and the day of the week(weekday or holiday), and the system outputs the minimum time, the arriving time, and the route. The minimum time is the time from starting at the entrance to returning to the exit(User can also search the minimum time from starting at any attractions to returning to any attractions). Furthermore, the entrance and the exit are collocated in Tokyo Disneyland.
More than two attractions and less than eight attractions can be input to, this system, so visitors can search for routes for from two to eight attractions.

We present the following formula to calculate the time from the entrance to exit. (The following sign is defined when we present a formula.)

I : a set of attraction
T :a set of time zone
M_{ij} : transit time of from "attraction $i \in I$" to "attraction $j \in I$"
M_{kg} : transit time of from "attraction $k \in I$" to exit
M_{gh} : transit time of from entrance to "attraction $h \in I$"
W_{it} : waiting time of "attraction $i \in I$" when time is $t \in T$
P_i : seat-load time of "attraction $i \in I$"
A : a set of branch{ $(i,j) \mid i,j \in I$}

The following formula illustrates the minimum time.

・Formula considering transit time, waiting time, and seat-load time

$$M_{gh} + \sum_{(i,j)\in A}(M_{ij} + W_{it} + P_i) + M_{kg} \rightarrow \min$$

We next describe the system structure. First, we implemented the above calculation and timetable in ECLiPSe software. This program can be executed on an ECLiPSe terminal. The following figure depicts the execution screen. B in Results in Fig.2 is the minimum time, and the side of result(space - splash...) represents the route.

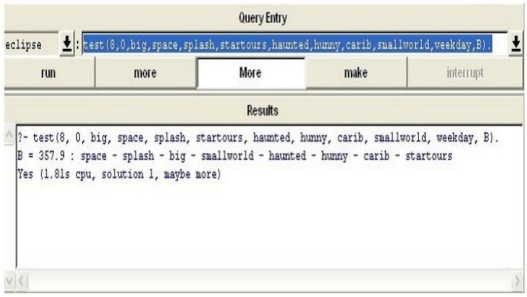

Figure 2. Execution Screen With ECLiPSe

System structure

We can construct a web application with this ECLiPSe program in two ways.

One way is to interact with Java. ECLiPSe can interact with Java [13] using a Java-ECLiPSe interface. The Java-ECLiPSe interface is a general-purpose tool for interfacing ECLiPSe with Java, Sun's popular object-oriented platform-independent programming language. For example, it could be used to write a Java graphical front-end for an ECLiPSe optimization program or to interface ECLiPSe with a commercial application written in Java.

The other way is to execute Java on the web with a Java Servlet. A Java Servlet is a program for dynamically generating an HTML document for a web page with Java. We construct a dynamic web site for a smart phone to use this function.

Figure 3 illustrate the structure of our system. The system incorporates Apache and Tomcat; Apache is used to build a web server, and Tomcat is used for the Java Servlet.

First, visitor accesses the web site we made with a smart phone. The visitor then inputs the starting time, weekday or holiday, and the eight attractions he/she would like to ride, and this information is passed to the Java program, and to the ECLiPSe program. The path that the ECLiPSe program should search for the minimum time must be described in the Java program.

The ECLiPSE program is then run on Java and calculates the route. The result is passed to the Java program as a type of object; therefore, we need to transform the object type into a string type. Because the string information contains the extra information, the character strings have to be split and the extra information have to be removed with Java's method. When the object is split, nine words are produced and stored in a list. The first of the nine words is the minimum route time. The remaining eight words are the attractions' names(the argument names in Table 1) stored in a list in the order of travel. For example, the first word in the list indicates the first attraction to which the visitor should travel, and the eighth word indicates the last attraction to be traveled to.

We note that this system is for Japanese people, and so on, the output needs to be translated into Japanese. The translation is performed in Java. In addition, the arrival time is also calculated in Java.

When the above process is completed, the system will output the total time, arrival time, and minimum route. Finally, the data is output on the web with the Java Servlet.

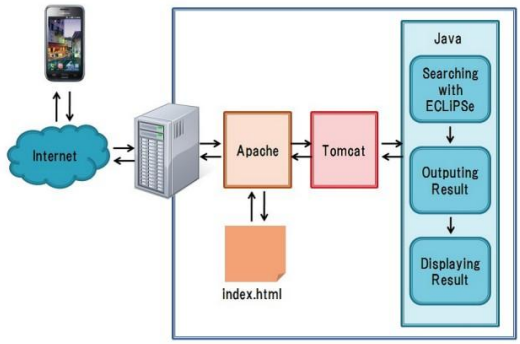

Figure 3. System Structure

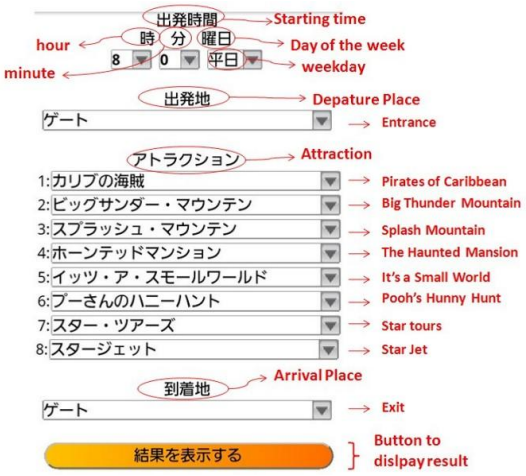

Figure 4. Top Screen

Interface

Our system has the following interfaces. Figure 4 depicts the top screen. This screen is displayed when a visitor first accesses our site. The visitor then inputs the starting time, weekday or holiday, and attractions from this screen. After two or three seconds, the result screen (Fig. 5) will display the starting time, arrival

time, weekday or holiday, the sum time, and the minimum route for attractions.

4. SYSTEM VERIFICATION

In this section, we demonstrate the effectiveness of our system.

Verification environment

The system employ hardware and software in Table 2.

Table 2
Hardware and Softwares to be used

CPU	Intel(R) Celeron(R) CPU 2.00GHz
Memory	1.03GB
OS	CentOS 5.3
ECLiPSe	ECLiPSe 6.0
Java	Jdk1.6.0_25
Tomcat	Tomcat 6.0
Apache	Apache 2.2.3

Results

Average and minimum times: We voluntarily select eight attractions and compare the sum time which our system calculated with the average time. This time is the average time to be calculated when the same transit time, waiting time, and seat-load time are used. (Someday we have to use this system in Tokyo Disney Land and confirm the accuracy of this system.)

Our system assumes that the starting time is 9 o'clock, and that the attractions selected are No.1 to No.8 in Table 1; the result is presented in Table 3.We also output the run time. We caluculated ten times and the average time was applied to run time. The run time in the case of weekday is 1.83s and in the case of holiday is 1.94s.

Discussion: Our system's time and the average time differ by 143.1 minutes on weekdays and 158.4 minutes on holidays in theory. These time differences illustrates the time reduction by using our system. Our system thus has the possibility to reduce the total travel time. Of course, a margin of error will be introduced because these results are calculated by simulation. However, we think that the results provided by our system will become significant when visitors search for the most suitable route. In conclusion, we have demonstrated that using our system shortens the total time.

Figure 5. Resylt Screen

Table 3. Total Time

		TIME	ROUTE
weekday	OUR SYSTEM	357.9(min)	2-1-5-7-4-6-3-8
	AVERAGE	501.0(min)	5-2-8-6-7-3-4-1
holiday	OUR SYSTEM	642.6(min)	2-7-1-3-5-4-6-8
	AVERAGE	801.0(min)	5-1-4-8-2-6-7-3

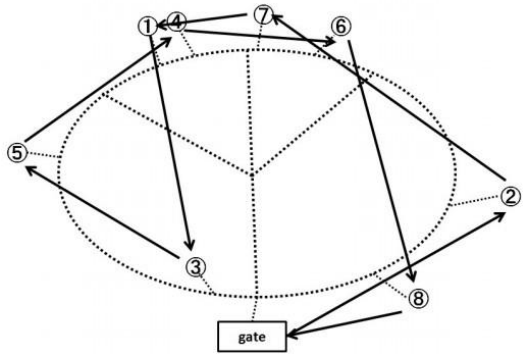

Figure 6. Route in the sum time calculated by our system

5. CONCLUSION

This study constructed a system for searching routes in Tokyo Disneyland using a smart phone and presented the system outline and interface. Additionally, we

compared the average total time with the time calculated by our system and concluded that our system is useful because it reduces the time required by about 150 minutes.

Currently, we are seeking to resolve three problems in this system. The first problem is that the number of attractions visitor can select is fixed in our system. We need to change fixed inputs to variable inputs. The second problem is that our system could introduce an increasing error. We have to recalculate the total time en route. (Though we have already completed the programs to solve these problems.) The last problem is that we didn't consider the Disney Fastpass, a virtual queuing system created by the Walt Disney Company. Fastpass enables visitors to avoid long lines at the attractions on which the system is installed, freeing them to enjoy other attractions during their wait. We completed the ECLiPSe program for calculating the minimum time when using Fastpass and now incorporate it into a smart phone. We will focus on the above three improvements in the future.

6. ACKNOWLEDGE

We thank the Research Institute for Science and Technology in Tokyo University of Science for financial support.

7. REFERENCE

[1] Yuki M., Hisayoshi, S. & Miho, T: OR of Amusement park ― as an example of The 2005 World Exposition, Aichi, Japan ― http://www.seto.nanzan-u.ac.jp/msie/grthesis/ms/2005/index.html

[2] Syouta, H., Yuka, H. & Daisuke, N: The minimum route in universal studio Japan - http://www.seto.nanzan-u.ac.jp/msie/grthesis/ms/2007/04mm010.pdf

[3] S, Lin. & B, W, Kernighan : An Effective Heuristic Algorithm for the Traveling-Salesman Problem. Operations Research 1973,Vol. 21, No. 2, pp. 498-516

[4] Stutzle, T. & Hoos, H. : Ant System and local search for the traveling salesman problem. Evolutionary Computation, 1997., IEEE International Conference, 1997, 309-314.

[5] Takashi, H., Takayuki, K. & Tohru, I : Solving vehicle routing problems with soft time windows using chaotic neurodynamics. IEICE transactions on fundamentals of electronics, communications and computer sciences 2006, 105(675), 17-22.

[6] Takahiro, S., Yoshihiro, H. & Koji N : Inversed Function Delayed Network for Traveling Salesman Problem. IEICE transactions on fundamentals of electronics, communications and computer sciences 2007, 107(328), 55-60.

[7] Takenori, M. & Naoki, M : Comparison of Approximate Methods for Traveling Salesman Problem. IEICE transactions on fundamentals of electronics, communications and computer sciences 2003, 102(625), 1-6.

[8] Eiichi, G., Etsuji, T. & Mitsuo, W : A Randomized and Genetic Hybrid Algorithm for the Traveling Salesman problem. Information Processing Society of Japan, 2001, 2001(27), 61-64.

[9] Caseau, Y. & F. Laburthe. : Solving small TSPs with constraints. In Proceedings the 14th International Conference on Logic Programming, 1997, 316-330

[10] http://www.mapion.co.jp/route/

[11] http://www.tokyodisneyresort.co.jp/tdl/

[12] http://www15.plala.or.jp/gcap/disney/

[13] http://www.eclipseclp.org/doc/embedding/embroot041.html