A practical route search system for amusement parks navigation

Takahiro Shibuya
Faculty of Science and Technology,
Tokyo University of Science,
2641 Yamazaki, Noda, Chiba, 278-8510, Japan

Masato Okada
Faculty of Science and Technology,
Tokyo University of Science,
2641 Yamazaki, Noda, Chiba, 278-8510, Japan

and

Hayato Ohwada
Faculty of Science and Technology,
Tokyo University of Science,
2641 Yamazaki, Noda, Chiba, 278-8510, Japan

ABSTRACT

It is very difficult to find the minimum route to travel in amusement park navigation. A searching system for visitors would be useful. Therefore, we constructed a system to find the route with the minimum total traveling time. Facility visitors can employ this system on a smart phone. The system is composed of Java and a Java Servlet. We conclude that our system is useful and can greatly shorten travel time within a typical amusement park.

Keywords: Traveling salesman problem, Traveling problem in amusement parks, Smart phone, Java Servlet.

1. INTRODUCTION

The many attractions in popular amusement parks make it difficult for visitors to find the fastest way of moving about. A visitor choosing a very slow route may become tired from walking and waiting, and may miss the opportunity to ride a desired attraction.

We believe 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 [1]. A traveling salesman must find the shortest possible route that visits each city exactly once, 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 a two-opt method and a simulated annealing method, which are among the meta-heuristics methods used to search for an approximate solution [2].

Research is also being conducted to propose how to travel efficiently with CPLEX [3].

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. ATTRACTION DESCRIPTION

Target

We constructed 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 cases where the service is suspended.

Location of Attraction

Figure 1 presents a map of Tokyo Disneyland, and Table 1 lists each attraction’s number, name, and type. The most popular attractions are marked in a separate column. This popularity is important for the present study, because our system focused on estimating waiting times for popular attractions.

The shortest distance between attractions is measured with "Kyorisoku" [4], a map service used to measure distance provided by Mapion Co., Ltd. (a Japanese company). When we measure distances, we use the map with Kyorisoku. We regard walking speed as three kilometers per hour and used this value for calculating the transit time.

We assumed 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 do not know the exact layout.

We adopted data from the official site of Tokyo Disneyland [5] for the seat-load time. Actual waiting times were published in May 2013 by Tokyo Disneyland. We adopted an off-season day and a very busy day, updating the waiting times every thirty minutes. We utilized the site "Congestion expectation calendar in Tokyo Disneyland" [6] to estimate the waiting time.

Table 1. Attractions in Tokyo Disneyland

ID	Name of attraction	Popular
1	Splash Mountain	○
2	Space Mountain	○
3	Prates of the Caribbean	
4	The Haunted Mansion	○
5	Big Thunder Mountain	○
6	Pooh's Hunny Hunt	○
7	It's a Small World	
8	Star Tours	○
9	Jungle Cruise	
10	Western River Railroad	
11	The Enchanted Tiki Room	
12	Westernland Shootin' Gallery	
13	Country Bear Theater	
14	Mark Twain Rivarboat	
15	Tom Sawyer Island Rafts	
16	Beaver Brothers Explorer Canoes	
17	Peter Pan's Flight	
18	Snow White's Scary Adventures	
19	Pinocchio's Daring Journey	
20	Dumbo The Flying Elephant	
21	King Arthur Carrousel	
22	Alice's Tea Party	
23	Roger Rabbit's Car Toon Spin	
24	Minnie's House	
25	Mickey's House and Meet Mickey	○
26	Gadget's Go Coaster	
27	Chip'n Dale's Treehouse	
28	Donald's Boat	
29	Buzz Lightyear	○
30	Star Jet	
31	Autopia	

Business hours actually differ by date, but we assumed that Tokyo Disneyland is open from 9 a.m. to 10 p.m.

3. SYSTEM OUTLINE

Program outline
In previous work, we constructed a system using a constraint logic programming framework that is useful for optimization, planning, scheduling resource allocation, timetabling, transport, etc. [7]. However, due to the system integration required for a real application, we created a search algorithm that was implementable within a conventional procedural language, specifically Java

A typical route search program for the traveling salesman problem is based on the shortest-path algorithm proposed by Dijkstra [8]. In this algorithm, the transit and waiting times are fixed. In contrast, amusement parks have variable waiting times, depending on the day of the week, holidays, and attraction popularity. This means that the minimum traveling time should be calculated within the waiting-time variation. We call this type of program a dynamic shortest-path program (DSP), as compared to a conventional shortest-path program (SP)

The visitor inputs the starting time, a list of attractions to be visited, and the date, used to identify season-off or busy days. The system outputs the minimum traveling time and the route. The minimum traveling time is the time from starting at the entrance to returning to the exit. The user can also determine the minimum traveling time starting at any attraction and

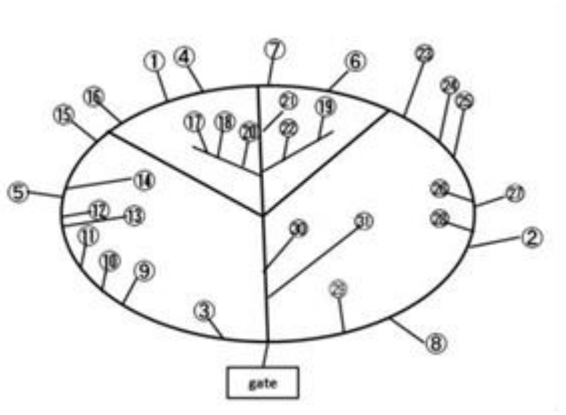

Fig. 1. Attraction map

returning to any attraction. As it happens, the entrance and the exit are collocated in Tokyo Disneyland.

We present the following formula to calculate the time from entrance to exit.

I: a set of attractions
T: a set of time zones
M_{ij}: transit time from "attraction $i \in I$" to "attraction $j \in I$"
M_{kg}: transit time from "attraction $k \in I$" to exit
M_{gh}: transit time from entrance to "attraction $h \in I$"
W_{it}: waiting time at "attraction $i \in I$" when time is $t \in T$
P_i: time of stay at "attraction $i \in I$"
A: a set of branches $\{ (i, j) \mid i, j \in I \}$

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

Figure 2 provides a DPS algorithm to find the minimum traveling time. The global variable MIN_PATH is introduced to store the tentative route in Line 2, and the global variable MIN_TIME to store its traveling time in Line 3. The search procedure is invoked with a set of attractions to be visited and a set of paths traveled so far in Line 6.

The search procedure is recursive. If there are no attractions to be visited, the procedure calculates a tentative travel time in Line 10 and updates MIN_PATH and MIN_TIME in Lines 11-13. Otherwise, an attraction is selected from REMAINING_LIST, intermediate information is calculated in Lines 17-20, and the search procedure is invoked recursively.

System structure
Our system is a web application with a Java Servlet. A Java Servlet is a program for dynamically generating an HTML document for a web page with Java. We constructed a dynamic web site to be accessed by a smart phone to use this function.

```
1.  PROCEDURE TRAVEL
2.   MIN_PATH ← ϕ;
3.   MIN_TIME ← ∞;
4.   INITIAL_LIST ← ATTRACTIONS;
5.   INITIAL_PATH ← <>;
6.   SEARCH(GOAL, INITIAL_LIST, 0, INITIAL_PATH);
7. END

8. PROCEDURE SEARCH(attraction, REMAIN_LIST, time, PATH)
9.   IF REMAIN_LIST is empty THEN
10.      traveling ← time + distance(attraction, GOAL);
11.      IF traveling < MIN_TIME THEN
12.         MIN_PATH ← PATH;
13.         MIN_TIME ← traveling;
14.      END IF
15. ELSE
16.    FOR each x in REMAIN_LIST
17.         traveling ← time + distance(attraction, x);
18.         getting_off ← traveling + wait(x, traveling) + seat-load(x);
19.         P_STACK ← PATH;
20.         add x to the end of P_STACK;
21.         L_STACK ← REMAIN_LIST-{x};
22.         SEARCH(x, L_STACK, getting_off, P_STACK);
23.      END FOR
24.   END IF
25. END
```

Fig. 2. Dynamic shortest-path algorithm

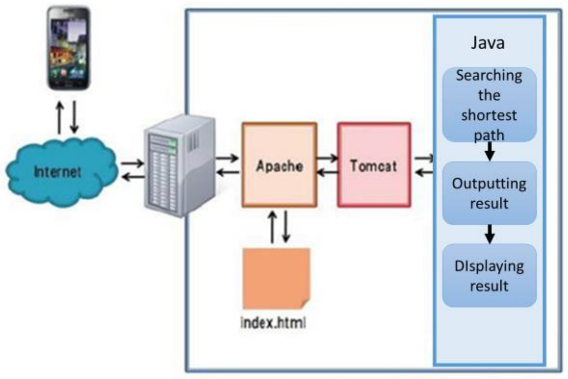

Fig. 3. System structure

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

First, a visitor accesses the web site with a smart phone. The visitor then inputs the starting time, the date, and at most eight attractions he/she would like to visit. This information is passed to the Java program.

Note that this system is for Japanese people, so the output needs to be translated into Japanese. The translation is performed in Java.

When the above process is completed, the system will output the total time, the traveling time, and the minimum route. Finally, the result is displayed using the Java Servlet.

Interface

Our system has the following interfaces. Figure 4 depicts the top-level interface. This is displayed when a visitor first

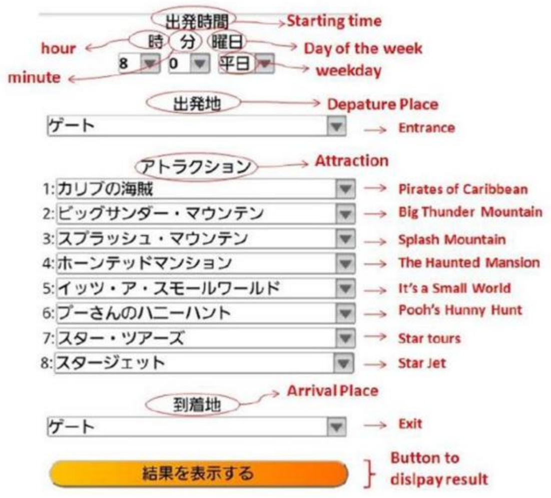

Fig. 4. Top-level interface

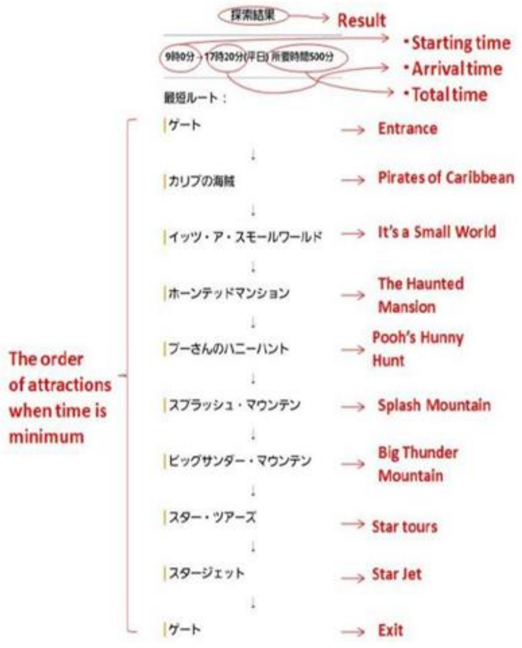

Fig. 5. Calculated information

accesses our site. The visitor then inputs the starting time, the date information, and the attractions to be visited. After two or three seconds, the calculated information, including the traveling time and the minimum route, will be displayed as seen in Fig. 5. Here, for the smart phone interface, at least two attractions, but no more than eight attractions, must be entered.

4. SYSTEM VERIFICATION

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

Verification environment

The system employs the hardware and software listed in Table 2.

Table 2. Hardware and software used

CPU	Intel Core i5 560M 2.56GHz
Memory	4.0 GB
OS	Windows 7 Pro
Java	JRE 1.7.0
Tomcat	Tomcat 6.0
Apache	Apache 2.2.3

Table 3. Traveling times and routes for popular attractions

Day	Method	Traveling time	Route
Normal day	DSP	293 min.	29-2-25-5-4-1-6-8
	SP	400 min.	5-1-4-6-2-25-8-29
Busy day	DSP	715 min.	1-5-2-4-6-25-29-8
	SP	827 min.	5-1-4-6-2-25-8-29

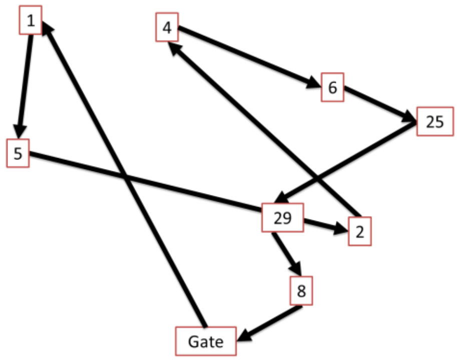

Fig. 6. Route for popular attractions calculated by DSP method on a busy day

Results

As a typical experiment setting, eight attractions are selected and applied to a normal day (weekday) and a busy day (holiday). The experiment measured the minimum traveling times and the runtimes for the DSP and SP programs.

Table 3 lists the minimum traveling times and routes for the eight popular attractions. DSP shortens the traveling times for both normal and busy days. The reduction in traveling time is quite effective, and the ranking of the route produced by SP is dramatically improved. The number of total routes is 40,320 for the eight-attraction permutation, and the order of the route determined by SP ranks 6940th for a normal day and 14,962th for a busy day. This is also observed from the routes found by DSP and SP. Figure 6 presents a minimum traveling route which is quite unlike the shortest path in Fig. 7. Thus, DSP suggests a reasonable route for efficiently visiting popular attractions.

Table 4 lists the minimum traveling times and routes for eight normal (i.e., less popular) attractions. The improvement in

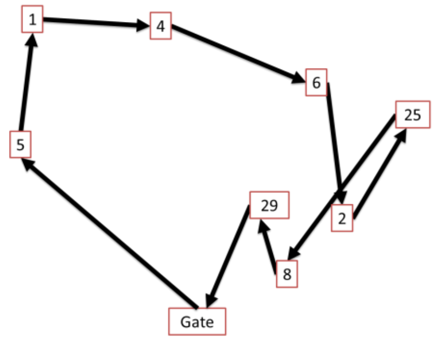

Fig. 7. Route of popular attractions calculated by SP method on a busy day

Table 4. Traveling times and routes for normal attractions

Day	Method	Traveling time	Route
Normal day	DSP	86 min.	11-15-21-7-22-24-27-28
	SP	86 min.	11-15-21-7-22-24-27-28
Busy day	DSP	121 min.	24-27-28-21-7-22-15-11
	SP	146 min.	11-15-21-7-22-24-27-28

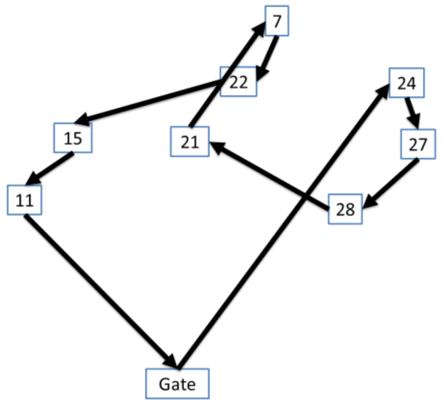

Fig. 8. Route for normal attractions calculated by DSP method on a busy day

traveling times due to DSP is small because their waiting times do not influence the total traveling time. Actually, DSP produced the top-ranked route for a normal day because the waiting times for the less popular attractions are zero. As for a busy day, the order of the route by SP is 4431th, with a small improvement in the traveling times. Figures 8 and 9 illustrate this situation.

Figure 10 presents the distributions of the minimum traveling times for a normal day; Fig. 11 presents those for a busy day. The total number of possible paths is 12,870, and the histograms are constructed by calculating the minimum traveling times produced by DSP (blue line) and DP (orange line). For both figures, DSP produced a shorter traveling time

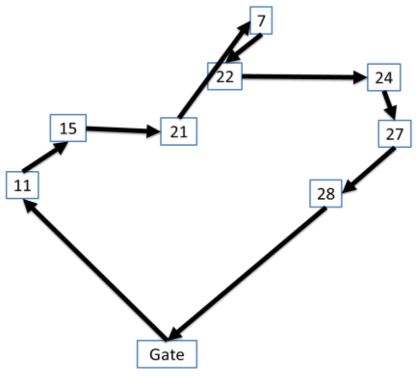

Fig. 9. Route for normal attractions calculated by SP method on a busy day

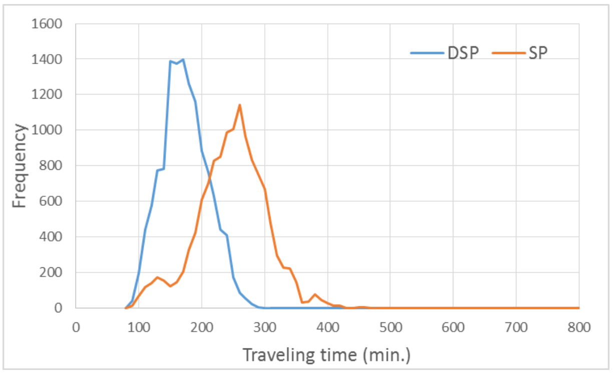

Fig. 10. Traveling-time distribution on a normal day

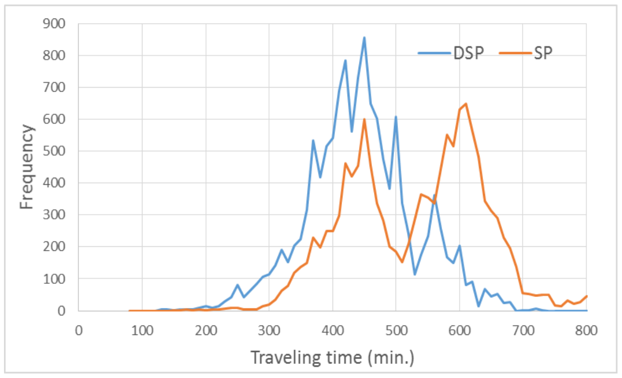

Fig. 11. Traveling-time distribution on a busy day

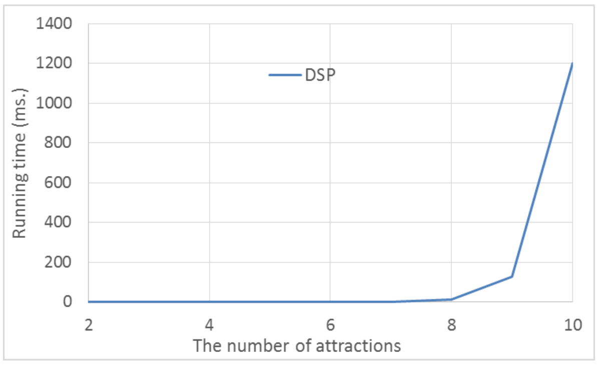

Fig. 12. Average running time

than that of SP. Note that oscillation occurs in the histogram on a busy day due to the variability of waiting times for each attraction.

We also calculated the runtime of our system, as seen in Fig. 12. The resulting performance is practical because most users specify at most eight attractions to be visited and requires 10 hours to visit popular attractions. Even in this extreme case, the running time to find the minimum traveling time is about 0.1 second for nine attractions.

5. CONCLUSION

This study constructed a system for searching routes at 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 100 minutes.

6. REFERENCES

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

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

[3] Syouta H., Yuka H. Daisuke N., T**he minimum route in universal studio Japan,** (in Japanese), Available: http://www.seto.nanzanu.ac.jp/msie/gr-thesis/ms/2007/04mm10.pdf

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

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

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

[7] Shibuya Takahiro, Ohwada Hayato, **A Constraint-Based Route Search System for Smart Phone in Attraction Facilities**, IMCIC 2012, pp. 111-115.

[8] Dijkstra, E.W., **A Note on Two Problems in Connexion with Graphs**, Numerische Mathematik, 1959, Vol. 1, pp. 269-271.