

Implementation for the Arrangement and Mining Analysis of Travel Schedules

YIN-FU HUANG AND CHAO-NAN CHEN

*Institute of Computer Science and Information Engineering
National Yunlin University of Science and Technology*

Touliu, 640 Taiwan

E-mail: huangyf@el.yuntech.edu.tw

In the paper, we use data mining techniques to analyze and retrieve customers' travel patterns from the database in a travel system built in the WWW environment. Our solution consists of two phases. First, we propose three algorithms embedded in the system to arrange the travel routes and accommodations for customers. Second, we use mining techniques to extract the most interesting knowledge contained in the database. This implementation provides more convenient services and more useful information than other traditional travel agencies.

Keywords: Internet services, data mining, sequential patterns, algorithms, travel schedules

1. INTRODUCTION

Due to the capability of collecting a huge amount of data in a database and the fast development of data mining techniques in recent years, an increasing number of applications based on data mining are being used in business and industry. They use data mining techniques to extract some important and complex knowledge hidden in a database, such as multimedia information on the Internet [14], path traversal patterns in a distributed information-providing environment [5], medical insurance files [13], and customer information from a shopping center or retail store [1, 9], etc. The paper by Standing *et al.* [10] explores the opportunities for travel agencies in relation to the World Wide Web. They conclude that the future of travel agents is under threat because of competition from on-line travel agents and from customers using the Internet to book travel arrangements in a do-it-yourself manner. Nevertheless, at present, the channels to get the travel information are still traditional, e.g., travel books, travel agencies, and airline companies. Even though customers can access travel information through Internet searches, the information is plain and not analyzed before retrieval. Besides, most related papers [6, 15] are not related to travel arrangement (or mining) algorithms.

Currently this kind of Internet service is offered by a few travel agencies such as American International Travel [3], Booking on Line [4], and Travel Major [12]. Basically, a travel website should contain data such as the travel information, domestic or international airline schedules, lodging, transportation, and passport/visa processing. Among them, the travel information is the most important part in the website and is accessed frequently by customers. It may include introductions to vacation destinations and arrange-

ments for accommodations. However, the information is static; that is, the travel schedule is fixed and cannot be altered. It is impossible for customers to arrange their own schedules since interactions between customers and the website are limited. The website can only act as an information provider and cannot get feedback from customers about what they need. Therefore, on the other hand, a travel agency cannot get customer market analyses or business strategies from the website.

To ameliorate these drawbacks of a traditional travel website, we implemented a travel mining system on the Internet to serve customers and travel agencies. In addition to providing basic travel information, the system allows customers to dynamically select destination regions, cities, and spots, and then arrange an itinerary automatically. In the itinerary, schedules, accommodations, expenses, and transportation are all considered. Also, to make use of feedback from customers, we apply mining techniques for sequential patterns [2, 5, 7, 8, 11] to analyze the customer market and efficiently integrate the results. Thus the system will be a more efficient and interactive tool for travel arrangements than others proposed before.

The paper is organized as follows. The system architecture of a travel mining system is given in section 2. Three algorithms ECT, EST, and ELM are proposed to arrange routes and accommodations in section 3. The technique for mining itineraries is given in section 4. Then the system implementation is presented in section 5. Finally we give conclusions in section 6.

2. SYSTEM ARCHITECTURE

As shown in Fig. 1, the travel mining system is composed of four types of components, user sites, a travel agency, a database, and a travel engine. Customers can log on to the system to request travel information through the Internet. Here customers can specify three different interactions with the system. First, personal customer information should be specified to serve as constraints in the mining process. Second, customers can select the destination regions, cities, and spots they want to visit. Then the system will send arranged itineraries back to customers. Last, customers can request travel-related information such as the distribution of routes and customer survey reports from the system. In addition to providing travel information to customers, such as introductions to travel spots, transportation information between cities or spots, and accommodation information, the travel agency itself can also request the analyses of customers from the system. All information collected from customers and the travel agency is stored in the database to be processed and analyzed by the travel engine. The travel engine is the kernel of the system. It not only helps customers make their itineraries, but also provides the travel agency travel with related analyses.

In the database, all travel information should be arranged to serve as useful data for making itineraries and mining travel patterns. Here the E/R model is adopted as the development tool to create the travel database. The travel database consists of four parts. First, the destination areas, cities and spots are described in Fig. 2. Second, transportation information about cities and spots is described in Fig. 3. Third, the accommodation information in only cities is described in Fig. 4. Fourth, personal information and arranged itineraries are integrated and described in Fig. 5.

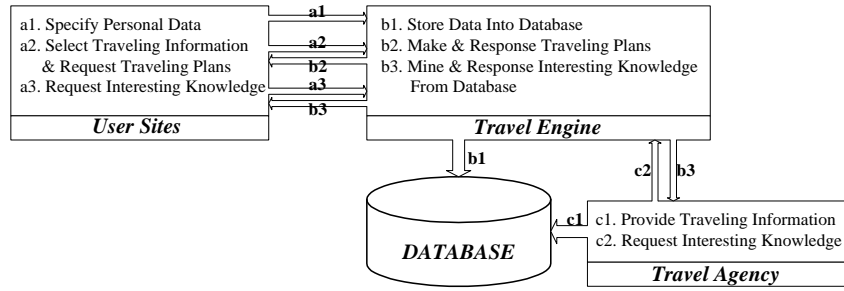


Fig. 1. Travel mining system.

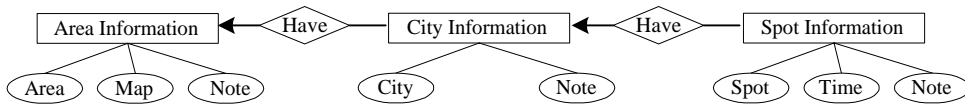


Fig. 2. Travel areas, cities, and spots.

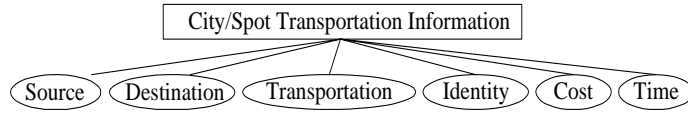


Fig. 3. Transportation information.

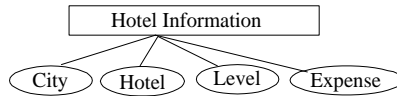


Fig. 4. Accommodation information.

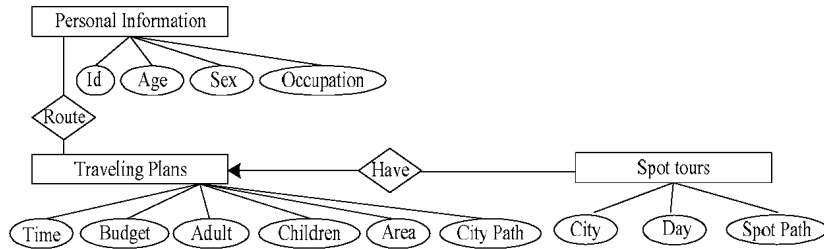


Fig. 5. Personal information and itineraries.

In the travel engine, we have four sub-problems to be solved. First, according to the cities specified by a customer the travel engine will find an efficient city route. The city route is arranged with a low cost (time or expense). Second, the travel engine will arrange a sequence of spot routes for each city to be visited based on the number of days the customer wants to stay. The spots are specified by the customer. Spot routes are similar to a

city route with respect to minimizing travel cost. Moreover, the spot routes are arranged to be “smooth-going” so that the spend time of each spot route is around the value suggested by the system. Third, to provide customers with more complete information, the travel engine also arranges accommodations for each day, including lodging and meals, according to their specified budgets. It also notifies customers if the arrangement is beyond their budgets. Fourth, the travel engine can find correlations between customer characteristics and travel arrangements proposed by the first two methods. The results include which city and spot routes are preferred by the specified customer and/or vice versa. The information can be referenced to adjust the future promotional activities of a travel agency.

3. ALGORITHMS

In this section, three algorithms embedded in the travel engine are proposed to arrange itineraries of customers. These algorithms are referred to as *Efficient City Tour* (ECT), *Efficient Spot Tour* (EST), and *Efficient Lodging & Meal* (ELM).

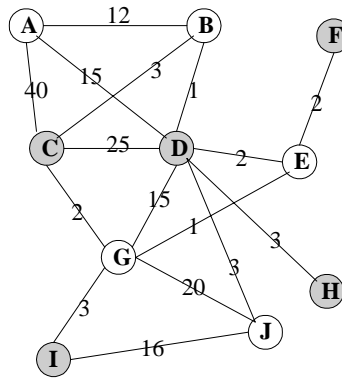


Fig. 6. City graph.

3.1 Algorithm: Efficient City Tour (ECT)

The goal of ECT is to find an efficient city tour according to the cities that the tourists would like to visit. An efficient city tour is a cost-effective way to visit all selected cities. The problem can be formalized as a graph problem. As shown in Fig. 6, cities are represented as nodes, transportation between cities are represented as undirected arcs labeled with the travel time. The shaded nodes denote the cities to be visited. One node is designated as a starting city from which a shortest tour to visit all shaded nodes is to be found. In general, a longer transportation time implies a greater transportation cost. Thus, we try to find the most efficient (or travel time effective) path in the ECT algorithm. This problem is different from the traveling salesperson problem. The goal of the traveling salesperson problem is to find a cycle tour with the minimum cost and includes all cities in a graph. However, the goal of our problem is to find a cost-effective tour but including

only selected cities. Further, the starting city in the tour should not be the ending city. Since the city tour problem is more likely an NP-complete problem, we prefer adopting a heuristic algorithm to solve it. Fortunately, the solution of the ECT algorithm is so convincing. We compared its results with the optimal results for 100 cases. We found that 87 ECT solutions are optimal solutions.

In brief, the procedure of the algorithm is as follows. The first step is to find all shortest paths from the starting node to the other selected nodes. The second step is to select an efficient path, with the minimum average cost, from among them. The third step is to designate the last node in the efficient path to be a new starting node. The last step repeats the procedure until all selected nodes are included in the concatenated path. To select an efficient path in step 2, we have two conditions for filtering out all useless shortest paths. For condition 1, we delete a path if it is a sub-path of any other paths since a path will include more selected nodes than its sub-path. For condition 2, we delete a path if all nodes in the path have been included in previous iteration, since the path is of no further use. Moreover, after filtering out all useless shortest paths, we still need a formula to calculate which shortest path has the minimum average cost, and choose it as the efficient path. The average cost of a shortest path can be formulated as

$$\text{average cost (shortest path)} = \frac{\text{the cost of a shortest path}}{\text{the number of selected nodes new-included}}.$$

The cost formula is used in each round for determining a sub-path, which has the minimum average cost to reach each selected city in the sub-path. Each sub-path determined in each round is obviously an optimal path. Within the ECT algorithm, each round is dominated by finding all the shortest paths from the starting node to other selected nodes using Dijkstra's algorithm. For a city graph (V, E) , since the time complexity of Dijkstra's algorithm is $O(V^2)$ and at most $|V|$ rounds are required to build the efficient path, the time complexity of the ECT algorithm is $O(V^3)$.

The formal algorithm for ECT is as follows:

```

/* T: the set of selected nodes,
   Snode: the starting node,
   Enodes: the set of selected nodes other than the starting node,
   Efficient_Path: a close-to-minimum path including all selected nodes,
   Efficient_Subpath: sub-path of a close-to-minimum path,
   S: the set of all shortest paths from Snode to the nodes in Enodes */
Efficient_Path :=  $\phi$ ;
Efficient_Subpath :=  $\phi$ ;
S :=  $\phi$ ;
Do
{
  Find all shortest paths from Snode to the nodes in Enodes using Dijkstra's
  Algorithm, and add them to S;
  /* Filter all useless paths */

```

```

For each path  $p \in S$ 
  If ( $p$  is the subset of any path in  $S$  or no nodes in  $p$  exist in  $T$ )
    then delete  $p$  from  $S$ ;
Endfor;
Efficient_Subpath = the path with the minimum average cost in  $S$ ;
/* Delete the nodes on Efficient Subpath from  $T$  */
For each node  $v$  on Efficient_Subpath
  If ( $v \in T$ )
    then delete  $v$  from  $T$ ;
Endfor;
Efficient_Path = Efficient_Path  $\cup$  Efficient_Subpath;
Snode = the last node in Efficient_Path;
Enodes = Enodes - Snode;
Efficient_Subpath :=  $\emptyset$ ;
S :=  $\emptyset$ ;
} While ( $T$  is not empty);

```

Here we take the city graph shown in Fig. 6 as an example, where five cities C, D, F, H, I are selected by a customer and node C is the starting city. The partial results for each round are as follows:

Round 1

Snode: C	Enodes: {DFHI}	$T = \{CDFHI\}$
----------	----------------	-----------------

S	average cost
$C \rightarrow B \rightarrow D$	discarded
$C \rightarrow G \rightarrow E \rightarrow F$	$5/2 = 2.5$
$C \rightarrow B \rightarrow D \rightarrow H$	$7/3 = 2.33$
$C \rightarrow G \rightarrow I$	$5/2 = 2.5$

$(C \rightarrow B \rightarrow D) \subseteq (C \rightarrow B \rightarrow D \rightarrow H)$, so discard $(C \rightarrow B \rightarrow D)$

$(C \rightarrow B \rightarrow D \rightarrow H)$ is an efficient subpath with the minimum average cost

Efficient_Path = $(C \rightarrow B \rightarrow D \rightarrow H)$

Round 2

Snode: H	Enodes: {DFI}	$T = \{FI\}$
----------	---------------	--------------

S	average cost
$H \rightarrow D$	discarded
$H \rightarrow D \rightarrow E \rightarrow F$	$7/1 = 7$
$H \rightarrow D \rightarrow E \rightarrow G \rightarrow I$	$9/1 = 9$

$(H \rightarrow D) \subseteq (H \rightarrow E \rightarrow E \rightarrow F)$, so discard $(H \rightarrow D)$

$(H \rightarrow D \rightarrow E \rightarrow F)$ is an efficient subpath with the minimum average cost

Efficient_Path = $(C \rightarrow B \rightarrow D \rightarrow H \rightarrow D \rightarrow E \rightarrow F)$

Round 3

Snode: F	Enodes: {DI}	$T = \{I\}$
----------	--------------	-------------

S	average cost
F → E → D	discarded
F → E → G → I	$6/1 = 6$

No nodes in (F → E → D) exist in T , so discard (F → E → D)

(F → E → G → I) is an efficient subpath

Efficient_Path = (C → B → D → H → D → E → F → E → G → I)

T is empty, so stop

3.2 Algorithm: Efficient Spot Tour (EST)

Similar to the problem of finding an efficient city tour, the problem explored here is to find an efficient spot tour given the spots and the dates specified by the tourists. The efficient spot tour should be cost-effective to visit all selected spots, and the time spend for each day should be balanced. The problem could be also formalized as a graph problem. As shown in Fig. 7, the spots around a city are represented as the nodes and the transportation between the spots are denoted by an undirected arc labeled with the transportation times. The weights labeled on the nodes indicate the suggested visiting times for each spot. The shaded nodes denote the spots to be visited during a spot tour. The node without a weight is the hotel where the tourists stay, and from which a shortest tour to visit all shaded spots is to be found. The tourists depart from the hotel each morning, visit the selected spots along a daily tour, and come back to the hotel in the evening. The next day they continue to visit the other selected spots until all selected spots around the city are visited.

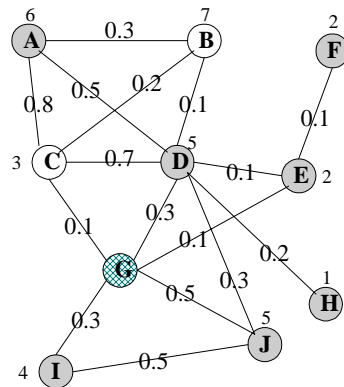


Fig. 7. Spot graph.

Here one constraint relative to the problem is given; that is the time spent visiting all selected spots should be less than or equal to eight hours times the number of days. The

reason is that a travel agency always schedules about eight hours for a daily tour. The constraint will help tourists to visit all the selected spots during their stays.

The ECT and EST problems seem similar to each other. Although they both use the shortest-path algorithm, their results are totally different. The ECT algorithm produces a single city tour path, while the EST algorithm produces multiple spot tour paths. Also, the computations for the average costs of both paths are completely different. Moreover, there are also some constraints imposed on the EST algorithm.

The algorithm arranges the selected spots for each daily tour. The first step is to exclude all illegal specifications about the number of selected spots and days, such as the constraint mentioned before. In the second step, if the number of selected spots is equal to the number of days, it is trivial just to arrange one spot for each daily tour. The third step is to find m shortest paths with smaller average cost from the starting node to the other selected nodes. Among the m shortest paths, no one is the “sub-path” of the others. The average cost of a path $u \rightarrow \dots \rightarrow v$ (i.e., $u \rightarrow *v$) can be computed as:

$$\text{Average_cost}(u \rightarrow *v) = \frac{\sum_{\substack{\forall \text{ selected nodes } k \text{ in path} \\ \text{and } k \neq u}} \text{weight}(k) + \text{Shortest_cost}(u, v)}{(\text{number of selected nodes in path}) - 1}.$$

This is similar to that for the ECT algorithm in concept. However, this is for determining the initial m sub-paths which have the minimum average cost for reaching each selected spot in the sub-paths. These initial sub-paths are obviously optimal paths. In the last step, we append the other selected nodes to the m shortest paths and select a “cost-effective” path from among them. This step repeats until all selected nodes are arranged.

The formal algorithm for EST is as follows:

```

/* T: the set of selected nodes,
  Snode: the starting node,
  m: the number of days for a stay,
  weight(v): the visiting time for node v,
  S: the set of paths,
  u → *v: the path from node (or sub-path) u to node v, i.e., u → ... → v,
  Last(p): the last node in path p,
  Distinctive_cost(p): the distinctive cost of path p, used in path concatenation,
  Cost(p): the total cost of path p,
  Average_cost(p): the average cost of path p, not including weight (starting node) */
/* Upper bound constraint */
If (  $\sum_{v \in T} \text{weight}(v) > 8 * m$  or  $m > |T|$  )
then Output “Illegal specification”; Stop;
If (  $m = |T|$  )
then Arrange one node for each daily tour; Stop;

```

```

Find all shortest paths from Snode to the nodes in  $T$  using Dijkstra's Algorithm,
/* Find  $m$  shortest paths with smaller average cost */
Do
{
   $S := \emptyset$ ;
  loop := false;
  Calculate the average cost of all shortest paths and add  $m$  shortest paths with
  smallest average costs to  $S$ ;
  For each path  $p \in S$ 
    If ( $p$  is the super-path of any path in  $S$ )
      then  $\forall$  overlapped nodes  $v$  in  $p$ ,  $\text{weight}(v) := 0$ ; loop := true;
  End for;
} While (loop = true);
For each path  $p \in S$ 
  For each node  $v$  in  $p$ 
    Delete  $v$  from  $T$ ;
  End for;
End for;
/* Append the other selected nodes in  $T$  to the shortest paths in  $S$  */
While ( $T$  is not empty) Do
{
  /* Calculate the distinctive cost when a node in  $T$  is appended to a path in  $S$  */
  For each path  $p \in S$ 
    For each node  $v \in T$ 
       $\text{Distinctive\_cost}(p \rightarrow *v) = (\text{Cost}(p) + \text{Average\_cost}(\text{Last}(p) \rightarrow *v))$ ;
    End for;
  End for;
  Select path  $p$  and node  $v$  with the minimum distinctive cost, say  $(p_{\min} \rightarrow *v_{\min})$ ;
  Append the sub-path  $\text{Last}(p_{\min}) \rightarrow *v_{\min}$  to the path  $p_{\min}$  in  $S$ ;
  Delete all selected nodes in the sub-path  $\text{Last}(p_{\min}) \rightarrow *v_{\min}$  from  $T$ ;
};

```

Here we take the spot graph as shown in Fig. 7 as an example, where seven spots A, D, E, F, H, I, J are selected by customers and they will be visited during a four day stay at hotel G. The partial results for each step are as follows:

Step 1:

$$G \xrightarrow{0.1} E(2) \Rightarrow 0+(2.1/1) = 2.1$$

$$G \xrightarrow{0.3} I(4) \Rightarrow 0+(4.3/1) = 4.3$$

$$G \xrightarrow{0.6} A(6) \Rightarrow 0+(6.6/1) = 6.6$$

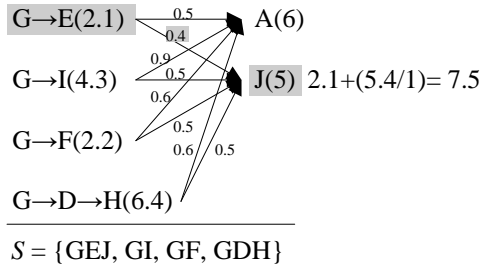
$$G \xrightarrow{0.1} E(2) \xrightarrow{0.1} F(2) \Rightarrow 0+(4.2/2) = 2.1$$

$$\begin{aligned}
 G \xrightarrow{0.1} E(2) \xrightarrow{0.1} D(5) &\Rightarrow 0+(7.2/2) = 3.6 \\
 G \xrightarrow{0.1} E(2) \xrightarrow{0.1} D(5) \xrightarrow{0.3} J(5) &\Rightarrow 0+(12.5/3) = 4.16 \\
 G \xrightarrow{0.1} E(2) \xrightarrow{0.1} D(5) \xrightarrow{0.2} H(1) &\Rightarrow 0+(8.4/3) = 2.8 \\
 \hline
 S &= \{GE, GEF, GED, GEDH\}
 \end{aligned}$$

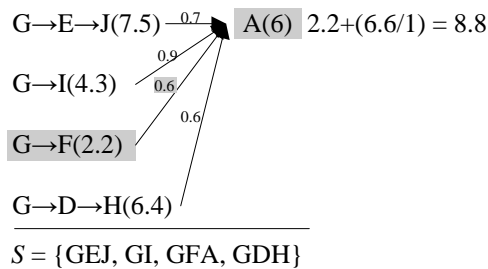
Step 2: Since $G \rightarrow E$ is a sub-path of some paths, their average costs should be re-calculated.

$$\begin{aligned}
 G \xrightarrow{0.1} E(2) &\Rightarrow 0+(2.1/1) = 2.1 \\
 G \xrightarrow{0.3} I(4) &\Rightarrow 0+(4.3/1) = 4.3 \\
 G \xrightarrow{0.6} A(6) &\Rightarrow 0+(6.6/1) = 6.6 \\
 G \xrightarrow{0.2} F(2) &\Rightarrow 0+(2.2/1) = 2.2 \\
 G \xrightarrow{0.2} D(5) &\Rightarrow 0+(5.2/1) = 5.2 \\
 G \xrightarrow{0.2} D(5) \xrightarrow{0.3} J(5) &\Rightarrow 0+(10.5/2) = 5.25 \\
 G \xrightarrow{0.2} D(5) \xrightarrow{0.2} H(1) &\Rightarrow 0+(6.4/2) = 3.2 \\
 \hline
 S &= \{GE, GI, GF, GDH\}
 \end{aligned}$$

Step 3: After appending the remaining spots A and J to the shortest paths in S, we find that the path $G \rightarrow E \rightarrow J$ has the minimum distinctive cost.



Step 4: After appending the last spot A to the shortest paths in S, we find that the path $G \rightarrow F \rightarrow A$ has the minimum distinctive cost.



3.3 Algorithm: Efficient Lodging & Meals (ELM)

In addition to arranging a city tour and a spot tour around a city, the system also does an efficient lodging and meal plan for an entire itinerary. An efficient lodging and meal plan is defined to be the most enjoyable for lodging and meals, and the total cost should be below the remaining budget after deducting the fixed cost such as fares and entrance fees, from the specified budget. As shown in Table 1, the hotels in a city can be classified into five levels and the meals can be classified into four. The algorithm chooses one combination of lodging and meals so that the cost is the most suitable.

Table 1. Classification of lodging and meals.

Level	Lodging		Meals	
	ratio	expense	ratio	expense
1	1.0	3000	1.0	2000
2	0.8	2400	0.5	1000
3	0.6	1800	0.25	500
4	0.4	1200	0.1	200
5	0.2	600		

The notation used in ELM algorithm is explained as follows:

- B remaining budget
- i city index
- n number of cities to be visited
- Day_i days spent in City $_i$
- R_i expense ratio in City $_i$
- E_i maximum expense of each day in City $_i$
- L_i highest level of lodging in City $_i$
- M_i highest level of meals in City $_i$

The expense ratio in City $_i$ is

$$R_i = \frac{Day_i * (L_i + M_i)}{\sum_{j=1}^n Day_j * (L_j + M_j)} \quad \forall i \in n$$

The maximum expense of each day in City $_i$ is

$$E_i = \frac{R_i * B}{Day_i} \quad \forall i \in n$$

Then the maximum expense of each day is compared with the lower bound that is the least expense of one day. If it is higher than the lower bound, we will find one com-

bination of lodging and meals with the highest expense. If more than two combinations have the same expense, we will choose the combination with higher expense of lodging. The algorithm is so simple that we omit it here.

Here we take an example to explain the algorithm. Given three cities C1, C2, and C3 to be visited by customers who will stay four days in C1, two days in C2, and four days in C3. The highest ratios of lodging and meals in C1 are 0.8 and 0.5, in C2 are 0.6 and 0.25, and in C3 are 0.6 and 0.5. For the remaining budget of \$20,000, the lodging and meals can be arranged as follows:

Step 1: /* Calculate the expense ratios */

$$R_{C1} = \frac{4(0.8+0.5)}{4(0.8+0.5) + 2(0.6+0.25) + 4(0.6+0.5)} = 0.46$$

$$R_{C2} = \frac{2(0.6+0.25)}{4(0.8+0.5) + 2(0.6+0.25) + 4(0.6+0.5)} = 0.15$$

$$R_{C3} = \frac{4(0.6+0.5)}{4(0.8+0.5) + 2(0.6+0.25) + 4(0.6+0.5)} = 0.39$$

Step 2: /* Calculate the maximum expenses for each day */

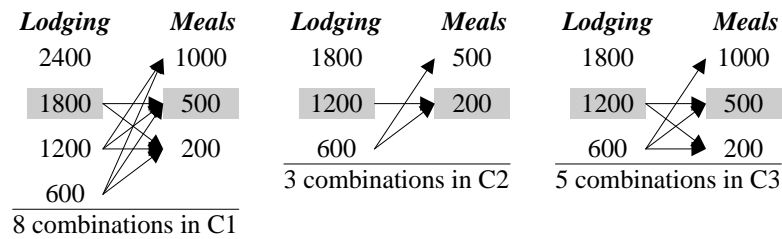
$$\text{Lower Bound} = 600 + 200 = 800$$

$$E_{C1} = \frac{0.46 * 20000}{4} = 2300 > \text{lower bound}$$

$$E_{C2} = \frac{0.15 * 20000}{2} = 1500 > \text{lower bound}$$

$$E_{C3} = \frac{0.39 * 20000}{4} = 1950 > \text{lower bound}$$

Step 3: /* Find one combination with the highest expense */



In C1, eight combinations fit the maximum expense of \$2300. We choose the combination with the highest expense. Finally the expenses of lodging and meals of each day in C1 are \$1800 and \$500, in C2 are \$1200 and \$200, and in C3 are \$1200 and \$500.

4. MINING ITINERARIES

The travel areas selected by customers and the itineraries arranged for them, including city and spot tours, have been stored in the database. Through data mining techniques, meaningful information can be extracted from the database to help a travel agency analyze the travel market. This information includes 1) the most popular area and tours for a specified group of customers, 2) which group of customers likes a specified area, and 3) the most popular city and spot tours of an area. In the following subsections, the sequence of travel routes is defined first. Then the search for the sequence patterns, including the generation of candidate sequences and the removing of sequences according to the minimum support criterion, is described.

4.1 Travel Sequences

Definition 1 Let $i = \{i_1, i_2, \dots, i_m\}$ be a set of literals, called *items*. An *itemset* is a non-empty set of items. A sequence is an ordered list of itemsets. Let s denote $\langle s_1 s_2 \dots s_n \rangle$, where s_j is an itemset. We also call s_j an *element* of the sequence. Let (x_1, x_2, \dots, x_m) denote an element of a sequence, where x_j is an item [2]. An item can occur only once in different elements of a sequence.

In the system, three sets of sequences, called travel areas, city routes, and spot routes, are associated with the data mining process. Since only one item is in an area sequence, a simple SQL statement will serve to mine out a meaningful area pattern and no complex mining process is required. For the city sequence and the spot sequence arranged by algorithm ECT and algorithm EST, though, a complex data mining process is required to extract useful sequence patterns. For the convenience of the mining process, we assume that the sets of travel areas, city routes, and spot routes are mapped to a set of contiguous integers as shown in Fig. 8.

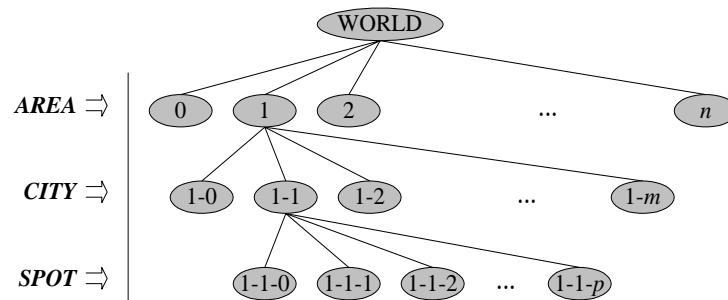


Fig. 8. Travel hierarchy.

4.2 Candidate Generation

Definition 2 Let k -sequences denote a sequence having k items. Let L_k represent the set of frequent k -sequences, and C_k the set of candidate k -sequences (potentially frequent

sequences). In each pass, k , the candidate k -sequences C_k are generated from the frequent $(k - 1)$ sequences L_{k-1} [1, 2]. The support for each item is the number of sequences that include the item.

Before generating candidates, we transform the original database table into the sequence table to facilitate the mining process. We take the spot tour as an example. The original database table is shown in Table 2, where the element (0-0-9, 0-0-5, 0-0-2) in the sequence of 890001 represents a spot tour on the first day (i.e., 0-0-9 \rightarrow 0-0-5 \rightarrow 0-0-2), and (0-0-8, 0-0-7) on the second day (i.e., 0-0-8 \rightarrow 0-0-7). The transformed results are shown in Table 3.

Table 2. Original database table.

Number	Spot tours
890001	$\langle (0-0-9, 0-0-5, 0-0-2) (0-0-8, 0-0-7) \rangle$
890002	$\langle (0-0-4, 0-0-7) (0-0-9, 0-0-5, 0-0-2) \rangle$
890003	$\langle (0-1-4, 0-1-7) (0-1-3, 0-1-9) \rangle$
890004	$\langle (0-0-9, 0-0-5) (0-1-3, 0-1-9) \rangle$
890005	$\langle (0-0-7, 0-0-9, 0-0-2) \rangle$
890006	$\langle (0-1-4, 0-1-7) (0-1-8, 0-1-3, 0-1-9) \rangle$

Table 3. Daily spot sequence table.

Number	Spot	Day	Position
890001	0-0-9	1	1
	0-0-5	1	2
	0-0-2	1	3
	0-0-8	2	1
	0-0-7	2	2
...

4.2.1 First pass of candidate generation

In the first pass of candidate generation, 1-sequence candidates can be found by scanning for each item throughout all sequences, and then we count the number of times (i.e., support) each item appears. If the support of one item is greater than or equal to the minimum support, then it is put into set L_1 . The results of the first pass are shown in Table 4 where column one is for 1-sequence candidates, column two is for the support, and column three is for associated information, such as customer ID, the day and position in a spot tour. For the first triple 0-0-2 in Table 4, the information (890001, 1, 3) means that customer id 890001 selects 0-0-2 as the third spot to be visited on the first day. If the minimum support is two for the first pass, then 1-sequence candidates 0-0-2, 0-0-5, 0-0-7, 0-0-9, 0-1-3, 0-1-4, 0-1-7, and 0-1-9 will be put into L_1 .

Table 4. 1-sequence candidates.

Spot tour	Support	(Cid, D, P)
0-0-2	3	(890001, 1, 3) (890002, 2, 3) (890005, 1, 3)
0-0-4	1	(890002, 1, 1)
0-0-5	3	(890001, 1, 2) (890002, 2, 2) (890004, 1, 2)
0-0-7	3	(890001, 2, 2) (890002, 1, 2) (890005, 1, 1)
0-0-8	1	(890001, 2, 1)
0-0-9	4	(890001, 1, 1) (890002, 2, 1) (890004, 1, 1) (890005, 1, 2)
0-1-3	3	(890003, 2, 1) (890004, 2, 1) (890006, 2, 2)
0-1-4	2	(890003, 1, 1) (890006, 1, 1)
0-1-7	2	(890003, 1, 2) (890006, 1, 2)
0-1-8	1	(890006, 2, 1)
0-1-9	3	(890003, 2, 2) (890004, 2, 2) (890006, 2, 3)

4.2.2 k th pass of candidate generation

Constraints in the next passes of candidate generation are as follows:

Join Constraint

In the k th pass, we generate k -sequence candidates by joining L_{k-1} with itself. The join predicate between $(k-1)$ -sequences S_1 and S_2 is that the subsequence obtained by dropping the first item of S_1 must be the same as the one obtained by dropping the last item of S_2 . The SQL statement can be expressed as follows:

```

insert into  $C_k$ 
select distinct  $I_1.item_1, I_1.item_2, \dots, I_1.item_{k-1}, I_2.item_{k-1}$ 
from  $L_{k-1} I_1, L_{k-1} I_2$ 
where  $I_1.item_2 = I_2.item_1, \dots, I_1.item_{k-1} = I_2.item_{k-2}$ 

```

Support Constraint

When two $(k-1)$ -sequences are joined together, if $(Cid1, D1, P1)$ for the first sequence and $(Cid2, D2, P2)$ for the second sequence satisfy the conditions that $Cid1 = Cid2$, $D1 = D2$, and $P2 = P1 + 1$, then the support is increased by one and a new entry $(Cid2, D2, P2)$ is inserted. For example, suppose two spot tours 0-0-9 and 0-0-2 are joined together. The corresponding information $(890005, 1, 2)$ and $(890005, 1, 3)$ satisfy the conditions, and the support is increased by one and the entry $(890005, 1, 3)$ is inserted for the new joined sequence.

Stop Constraint

Keep on generating frequent k -sequences until no new one can be found.

The second pass of candidate generation is shown in Table 5 where four frequent 2-sequences (0-0-5 \rightarrow 0-0-2), (0-0-9 \rightarrow 0-0-5), (0-1-3 \rightarrow 0-1-9) and (0-1-4 \rightarrow 0-1-7) satisfy the minimum support. In the third pass, only one frequent 3-sequence (0-0-9 \rightarrow 0-0-5 \rightarrow 0-0-2) is generated, as shown in Table 6. In the last pass, no 4-sequence candidate can be found and we terminate the candidate generation. Thus the frequent spot tour (i.e., sequential pattern) for customers from 890001 to 890006 is (0-0-9 \rightarrow 0-0-5 \rightarrow 0-0-2).

Table 5. 2-sequence candidates.

Spot tour	Support	(Cid, D, P)
0-0-4 \rightarrow 0-0-7	1	(890002, 1, 2)
0-0-5 \rightarrow 0-0-2	2	(890001, 1, 3) (890002, 2, 3)
0-0-7 \rightarrow 0-0-9	1	(890005, 1, 2)
0-0-8 \rightarrow 0-0-7	1	(890001, 2, 2)
0-0-9 \rightarrow 0-0-2	1	(890005, 1, 3)
0-0-9 \rightarrow 0-0-5	3	(890001, 1, 2) (890002, 2, 2) (890004, 1, 2)
0-1-3 \rightarrow 0-1-9	3	(890003, 2, 2) (890004, 2, 2) (890006, 2, 3)
0-1-4 \rightarrow 0-1-7	2	(890003, 1, 2) (890006, 1, 2)
0-1-8 \rightarrow 0-1-3	1	(890006, 2, 2)

Table 6. 3-sequence candidates.

Spot tour	Support	(Cid, D, P)
0-0-7 \rightarrow 0-0-9 \rightarrow 0-0-2	1	(890005, 1, 3)
0-0-9 \rightarrow 0-0-5 \rightarrow 0-0-2	2	(890001, 1, 3) (890002, 2, 3)

4.3 Discussions

In the sequence mining, we usually search the maximum-length sequences which are frequent (i.e., their support count are larger than or equal to the specified minimum support). In section 4.2, we found the frequent spot tour (0-0-9 \rightarrow 0-0-5 \rightarrow 0-0-2). We also have two frequent spot tours (0-1-3 \rightarrow 0-1-9) and (0-1-4 \rightarrow 0-1-7) which are not the sub-paths of any frequent spot tour. These frequent spot tours can be used to notify travel agencies what are the most popular spot routes, and what travel trend of customers is if their characteristics are specified.

5. IMPLEMENTATION

5.1 The Development Environment

In the travel mining system, a platform equipped with 1.6GHz P4, 256 MB memory, and 40 GB hard disk and running Windows 2000 server service pack 4, acts as a travel

engine as shown in Fig. 9. Moreover, the travel information is stored as a database under the management of Visual FoxPro. All visual user interfaces are designed using the Dynamic HTML syntax. The functions within the travel engine are implemented using Active Server Pages 3.0 (ASP 3.0), Scripting Languages (VBScript and JScript), and Visual Basic. Through ActiveX Data Object (ADO) and SQL, we can search the target information in the database.

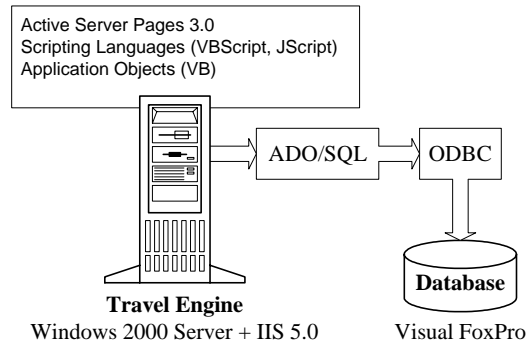


Fig. 9. Development environment for the travel engine.

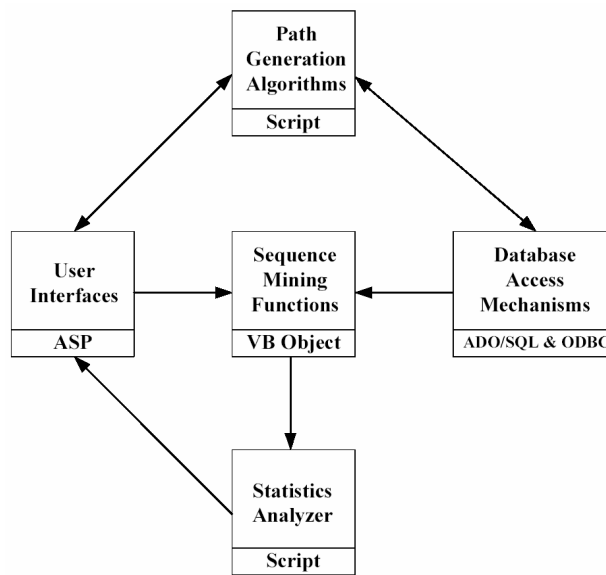


Fig. 10. Interoperable structure of functional components.

The functional components of the system consist of user interfaces, path generation algorithms, sequence mining functions, statistics analyzer, and database access mechanisms. Their interoperable structure is shown in Fig. 10. Customers can specify the per-

sonal information and demands through user interfaces, which are used to produce their itineraries by using path generation algorithms. At the same time, this information and the produced itineraries are stored in the database through database access mechanisms. Afterwards, customers or travel agencies can specify predicates through user interfaces to mine the information they desire. The source information is retrieved from the database through database access mechanisms. The mined results should be sent to the statistics analyzer before they are transmitted back to customers or travel agencies.

There are two phases to realize the system. In the first, we use artificial samples to validate the accuracy of each function, and then post travel homepages on the WWW to get the real samples of customers in the second phase. The artificial samples simulating 1000 customers include attributes such as age, sex, occupation, and their choices (e.g., areas, cities, spots, days, budgets) for itineraries. Although the customer information is generated at random, it still has some biases for mining frequent travel patterns. Thus, they are more or less reliable. As for the travel information, we have 10 travel areas, with at most 20 cities per area and at most 20 spots per city in the database.

5.2 Function Design and Visual User Interface

Four functions are provided in the system. They are the interface for managing itineraries, the interface for mining travel information based on customer characteristics, the interface for mining customer distribution based on travel information, and the interface for mining special requirements. Customers can access these functions via the main user interface as shown in Fig. 11.

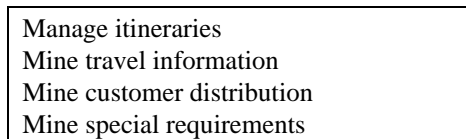


Fig. 11. Main user interface.

Interface for Managing Itineraries

The interface for managing itineraries is shown in Fig. 12 (a). Through the interface, a customer can specify his or her basic information and other information used to manage itineraries. After button GetCity is clicked, the map and the travel cities of the selected area are displayed as shown in Fig. 12 (b). Then the customer chooses the cities to be visited, and the system uses the ECT algorithm mentioned in section 3.1 to find an efficient city tour and dynamically plot the routes as shown in Fig. 12 (c). Next, as shown in Fig. 12 (d), the system lists all the spots around the selected cities, and the customer selects the spots to be visited and the number of days spent in each city. After the customer clicks Check, the system uses algorithms EST and ELM mentioned in section 3.2 and section 3.3 to find efficient spot tours and accommodations as shown in Fig. 12 (e). During the process, the customer will be asked to change his or her selection if the estimated expenses are larger than the specified budget.

Please fill out the following information and press GetCity :

Age:

Sex: M F

Occupation:

Traveling Budget: (Exclusive of passports & living expenses)

Total Number of Travelers: Adults Children

Traveling Area:

Fig. 12. (a) Interface for specifying basic information.

Traveling Area : Germany



Please choose the major cities you want to visit :

Berlin Cologne Dresden Dusseldorf Frankfurt
 Hamburg Hannover Leipzig Munich Stuttgart

Fig. 12. (b) Interface for selecting travel cities.

The following is a map of the city tour arranged for you !

City Tour: Berlin→Leipzig→Frankfurt→Munich



Please choose the major spots you want to visit

Fig. 12. (c) Arrangement of the city tour.

Please choose the spots and the number of days you prepare to stay :

Spots in Berlin	Spots in Leipzig	Spots in Munich
<input checked="" type="checkbox"/> Schlos Charlottenburg	<input type="checkbox"/> Altes Rathaus	<input type="checkbox"/> Neue Pinakothek
<input checked="" type="checkbox"/> Antikensammlung	<input type="checkbox"/> Altes Waage	<input type="checkbox"/> Alte Pinakothek
<input type="checkbox"/> Kaiser-Wilhelm-Gedachtnis-Kirche	<input type="checkbox"/> Barthels Hof	<input type="checkbox"/> Stadtische Galerie im Lenbachhaus
<input checked="" type="checkbox"/> Europa-Center	<input type="checkbox"/> Bosehaus	<input type="checkbox"/> Die Staatliche Antikensammlungen
<input type="checkbox"/> Zoologischer Garten	<input type="checkbox"/> Gohliser Schlosschen	<input type="checkbox"/> Karlstor
<input checked="" type="checkbox"/> Victory Column	<input type="checkbox"/> Madler-Passage	<input type="checkbox"/> Das Neue Rathaus
<input type="checkbox"/> Philharmonie	<input type="checkbox"/> Neues Rathaus	<input checked="" type="checkbox"/> Frauenkirche
<input type="checkbox"/> Berliner Mauer	<input checked="" type="checkbox"/> Nicolakirche	<input type="checkbox"/> Marienplatz
<input type="checkbox"/> Reichstagsgebäude	<input type="checkbox"/> Schillerhaus	<input checked="" type="checkbox"/> Die Kirche St. Peter
<input checked="" type="checkbox"/> Brandenburger Tor	<input checked="" type="checkbox"/> Thomaskirche	<input type="checkbox"/> Viktualienmarkt
<input checked="" type="checkbox"/> Museum Island	<input type="checkbox"/> Zum Coffe Baum	<input checked="" type="checkbox"/> Holfbrauhaus
<input type="checkbox"/> Neue Wache	<input type="checkbox"/> Grassi Museum	<input type="checkbox"/> National Theater
<input type="checkbox"/> Gendarmen Market	<input type="checkbox"/> Round Corner	<input type="checkbox"/> Residenz
<input type="checkbox"/> Berliner Dom	<input type="checkbox"/> Technikcenter	<input type="checkbox"/> Feldherrhalle
<input checked="" type="checkbox"/> Berlin Town Hall	<input checked="" type="checkbox"/> Moritzbastel	<input type="checkbox"/> Hofgarten
<input type="checkbox"/> Potsdamer Platz	Staying Days : <input type="text" value="1"/>	<input checked="" type="checkbox"/> Englisher Garten
Staying Days : <input type="text" value="3"/>		Staying Days : <input type="text" value="2"/> <input type="button" value="Check"/>

Fig. 12. (d) Interface for selecting travel spots and the number of staying days.

You stay in Berlin for 3 days, and your schedule is as follows :

Day1: Brandenburger Tor→Museum Island→Berlin Town Hall
Day2: Victory Column→Europa-Center
Day3: Antikensammlung→Schlos Charlottenburg
Lodging: Grand Hyatt Berlin
Meal Expenses: NT\$500

You stay in Leipzig for 1 day, and your schedule is as follows :

Day1: Nicolakirche→Thomaskirche→Moritzbastel
Lodging: Seaside Park Hotel Leipzig
Meal Expenses: NT\$800

You stay in Munich for 2 days, and your schedule is as follows :

Day1: Die Kirche St. Peter→Frauenkirche
Day2: Holfbrauhaus→Englisher Garten
Lodging: Hotel Splendid
Meal Expenses: \$600

Fig. 12. (e) Arrangement of the spot tours and accommodations.

Interface for Mining Travel Information

In the implementation, we can mine travel information based on the customer characteristics specified as shown in Fig. 13 (a). The travel information includes the most popular area, city route, and spot route as shown in Fig. 13 (b). These analyses results are useful for a travel agency to understand the travel trend of some specific customers.

Please select the Customer Attributes you want to analyze :

Age:

Sex: M F

Occupation:

Fig. 13. (a) Interface for mining travel information.

Analysis Results :

The most popular Traveling Area: Germany

The most popular City Route: Berlin→Hamburg

The most popular Spot Route: Berliner Mauer→Brandenburger Tor→Potsdamer Platz

Fig. 13. (b) Mining results based on customer characteristics.

Please select one Traveling Area you want to analyze :

Area:

Please select one Traveling City you want to analyze :

Area:

City:

Fig. 14. (a) Interface for mining customer distribution.

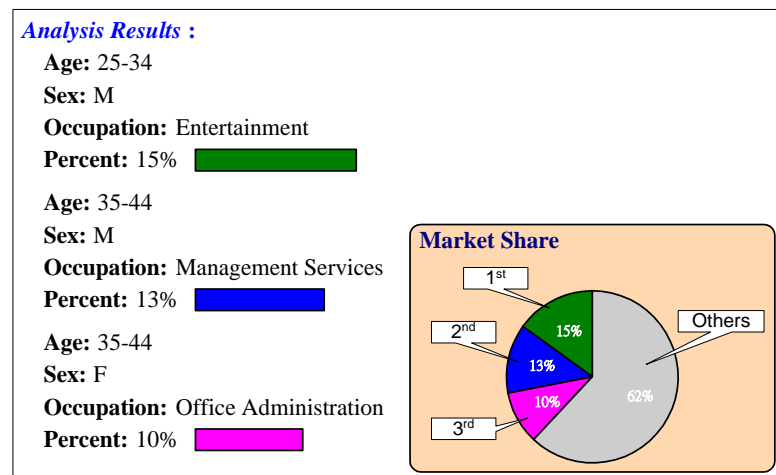


Fig. 14. (b) Mining results based on travel information.

Interface for Mining Customer Distribution

Unlike the interface for mining travel information based on customer characteristics, we can mine customer distribution based on the specified travel information as shown in Fig. 14 (a). The contents of customer distribution include the basic information and percentage of each group as shown in Fig. 14 (b). These analyses show which groups of customers like which specific areas (or cities).

Interface for Mining Special Requirements

In addition to the mining interfaces mentioned above, we can mine other travel information based on the special requirements specified as shown in Fig. 15 (a). This information includes the most popular travel area, city route, and spot route under various constraints, as shown in Fig. 15 (b).

*Please select the Item you want to analyze and press **SUBMIT** :*

The most popular Traveling Area in the World:

The most popular City & Spot Route in the World:

The most popular City & Spot Route in the Area:

The most popular City & Spot Route under the Budget:

Fig. 15. (a) Interface for mining special requirements.

Analysis Results :

The most popular City Route: Berlin→Hamburg

The most popular Spot Route: Brandenburger Tor→Potsdamer Platz

Fig. 15. (b) Mining results about the most popular routes.

6. CONCLUSIONS

We built a prototype travel mining system on the WWW not only to aid travelers in managing their itineraries, but also to help travel agencies analyze the travel market. Within the system, three embedded algorithms (ECT, EST, and ELM) are proposed to make the itineraries that will be collected in the database. Since the information contains some implicit customer preferences for routes, we can apply a data mining technique to reveal the meaningful knowledge that will serve as useful references for customers and travel agencies. Standing *et al.* propose [10] that there are various aspects of electronic commerce that can be exploited by travel agents. Some of these are listed below.

- (1) Use the Internet to build customer relationships – by having customers interact directly with the web sites;
- (2) Gather information from customers and potential customers to create customer profiles which can be used in marketing and product development;

- (3) Information partnerships – cooperation between organizations to provide better services to customers;
- (4) Transactions – reservations and bookings;
- (5) Provide specialized information based on the profile of a user;
- (6) Information and products that can be down-loaded by the user.

Among these aspects, our travel mining system is equipped with items 1, 2, and 5. Items 4 and 6, we believe, could be implemented without much difficulty in our system, although they have not been provided yet. We think item 3 is the most difficult job in electronic commerce, since cooperation needs time to develop between airlines, hotels, resorts, etc. Nevertheless, we believe that our travel mining system is a useful prototype system based on these aspects.

REFERENCES

1. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proceedings of 20th International Conference on Very Large Data Bases*, 1994, pp. 487-499.
2. R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proceedings of 11th IEEE International Conference on Data Engineering*, 1995, pp. 3-14.
3. American International Travel, <http://www.aitv.com/>.
4. Booking on Line, <http://www.bookingonline.com.tw/>.
5. M. S. Chen, S. P. Jong, and P. S. Yu, "Efficient data mining for path traversal patterns," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, 1998, pp. 209-221.
6. T. Feyer, O. Kao, K. D. Schewe, and B. Thalheim, "Design of data-intensive Web-based information services," in *Proceedings of 1st International Conference on Web Information Systems Engineering*, 2000, pp. 462-467.
7. M. Garofalakis, R. Rastogi, and K. Shim, "SPIRIT: sequential pattern mining with regular expression constraints," in *Proceedings of 25th International Conference on Very Large Data Bases*, 1999, pp. 223-234.
8. R. Srikant and R. Agrawal, "Mining sequential patterns: generalizations and performance improvements," in *Proceedings of 5th International Conference on Extending Database Technology*, 1996, pp. 3-17.
9. R. Srikant, Q. Vu, and R. Agrawal, "Mining association rules with item constraints," in *Proceedings of 3rd International Conference on Knowledge Discovery in Databases and Data Mining*, 1997, pp. 67-73.
10. C. Standing, S. Borbely, and T. Vasudavan, "A study of Web diffusion in travel agencies," in *Proceedings of 32nd Annual Hawaii International Conference on System Sciences*, 1999, pp. 1-9.
11. S. Thomas and S. Sarawagi, "Mining generalized association rules and sequential patterns using SQL queries," in *Proceedings of 4th International Conference on Knowledge Discovery in Databases and Data Mining*, 1998, pp. 344-348.
12. Travel Major, <http://www.travelmajor.com/>.
13. M. S. Viveros, J. P. Nearhos, and M. J. Rothman, "Applying data mining techniques

- to a health insurance information system,” in *Proceedings of 22nd International Conference on Very Large Data Bases*, 1996, pp. 286-293.
14. O. R. Zaïane, J. Han, Z. N. Li, and J. Hou, “Mining multimedia data,” in *Proceedings of Conference on the Centre for Advanced Studies (CASCON '98)*, 1998, pp. 83-96.
 15. J. Y. Zheng and M. Shi, “Mapping cityscapes to cyber space,” in *Proceedings of International Conference on Cyberworlds*, 2003, pp. 166-173.



Yin-Fu Huang (黃胤傳) received the B.S. degree in Computer Science from National Chiao Tung University in 1979, and the M.S. and Ph.D. degrees in Computer Science from National Tsing Hua University in 1984 and 1988, respectively. He is currently a Professor in the Department of Electronic Engineering, National Yunlin University of Science and Technology. Between July 1988 and July 1992, he was with Chung Shan Institute of Science and Technology as an Assistant Researcher. His research interests include database systems, multimedia systems, data mining, mobile computing, and bioinformatics.



Chao-Nan Chen (陳肇男) received his B.S. degree in Electronic Engineering and M.S. degree in Electronic and Information Engineering from National Yunlin University of Science and Technology in 1998 and 2000, respectively. He is currently a master student in the Department of Electrical and Computer Engineering, the University of Wisconsin-Madison. His major areas of interests are database systems, multimedia systems, computer networks, and digital circuit design.