Efficient Search Algorithms for Route Information Services of Direct and Connecting Transit Trips

ANTHONY F. HAN AND CHIEN-HUA HWANG

Easy-to-access and efficient route information service is essential to encourage the ridership of a transit system. Given the origin (bus) stop and destination stop of a transit trip, it is not difficult to find the direct bus lines connecting the given origin-destination stops. However, when the transit network structure is involved with many transfer trips or characterized with many overlapping bus lines, as are many transit systems in major cities outside North America, the problem of finding bus lines connecting through transfer points becomes very complicated. A simple yet efficient algorithm to find direct lines and a search algorithm using the hash-table data structure techniques for quickly finding routing information for connecting trips with one transfer are presented. The search algorithms have been successfully implemented on a microcomputer-based transit information service system in Taipei, Taiwan, since January 1991.

Easy-to-access network information is essential to encourage the ridership of a transit system. The more people who are aware of the transit network structure and routing information, the more people who can use the transit system. For this reason, most transit authorities in big cities worldwide are providing some form of information service to assist passengers in determining the best use of public transport for specific trips (*1–4*). Fruin described in detail the type of passenger information including visual and oral communication, distributed information, and automatic passenger interactive means (*5*).

The importance of using computerized management information technologies to improve the productivity and performance of a transit system has long been recognized. Recently, microcomputer-based systems have been widely applied to enhance the transit planning and operational capabilities in areas such as traffic and data management (*6*), fleet management (*7*), maintenance management (*8*), and performance monitoring (*9,10*). Cutler studied the impact of information technologies on the efficiency of telephone information services provided by 15 transit authorities in the United States (*11*). The success of such a telephone on-line transit routing information system relies on, among other factors, efficient search algorithms that can provide quick responses and accurate routing information. However, literature focused on the routing information search for direct and connecting transit trips appears to be rare. Vanigrok developed an algorithm for selecting desirable public transport connection from the time tables (*3*).

Department of Transportation Engineering and Management, National Chiao Tung University, Hsinchu 30049, Taiwan, Republic of China.

In this paper, we are concerned with the problem of finding all the bus lines connecting two bus stops with or without transfers in a transit network. For simplicity and pragmatic considerations, the following discussions will be limited to transit trips involving no more than one transfer. Given the origin (O) bus stop and the destination (D) stop of an inquired trip, the problem of finding the bus lines that directly connect the stops is not difficult if such lines exist. However, the problem becomes difficult when the bus transfer is inevitable. In particular, when the transit network is characterized with overlapping bus lines, as many transit systems in major cities outside North America are, the problem can be very complicated. For example, for the origin stop A and the destination stop B, shown in Figure 1, no direct bus lines connect A and B; the rider has seven transfer options: from Line 1 to Line 2, transferring at one of the stops $\{a, b, c, d\}$, or from Line 3 to Line 4, transferring at one of the stops $\{e, f, g\}$. Note that the situation is even more complicated when any one of the single lines (1, 2, 3, or 4) can be a set of overlapping bus lines running on the same street segment. As to the transit network characterized with heavily overlapping bus routes, Han and Wilson proposed a heuristic method for optimal bus allocation for the Cairo transit system (*12*), Han assessed the transfer penalty to bus riders in Taipei (*13*), and Kwan analyzed how to coordinate the joint headway for overlapping bus routes in the United Kingdom (*14*).

SEARCH ALGORITHM FOR DIRECT CONNECTING LINES

To provide accurate routing information, the system must distinguish the bus lines with different attributes such as direction and zonal or express service. As a result, almost every bus stop is covered by more than one bus line, say, 32 eastbound and 32 westbound. Consequently, when the transit network is characterized with heavily overlapping bus routes, finding direct connecting bus lines is much more complicated than expected. For example, in the Taipei Transit System there are approximately 490 bus lines and 1,200 bus stops, and more than 50 bus stops are served by more than 30 (overlapping) bus lines; the maximal overlapping stop is the Taipei Train Station, which is covered by 63 bus lines (*15*).

Given two bus stops, A and B, we now present a search algorithm to find all bus lines connecting directly from A to

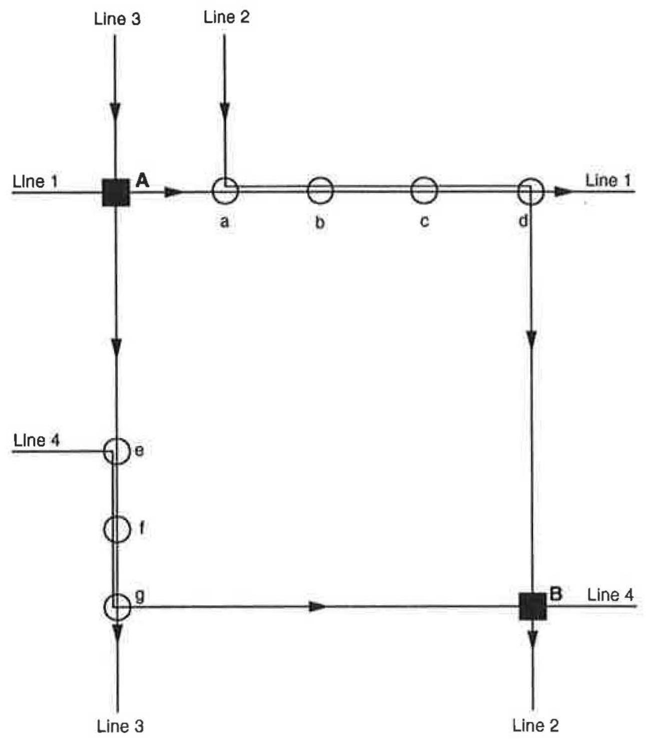


FIGURE 1 Connecting trip with multiple transfer stops.

B. The algorithm has two phases: connection search and direction check. The algorithm for direct transit trips is summarized in the following.

Algorithm A1: Connection Search Phase

1. Initialization: $S = \phi$, S is the set of direct connecting lines.
2. For both stops, *A* and *B*, list and sort all bus lines passing each of them. We get two arrays each presenting the set of bus lines passing the corresponding bus stop, and the numbers (codes) in both arrays are in ascending order.
3. Pick up from each of the two arrays the first number and do a pairwise comparison.
   - If the two numbers are the same—say, they are both N_1—then we have found a direct connecting line. Let $S = S \cup \{N_1\}$, and remove N_1 from both arrays.
   - If the two numbers are not the same, remove the smaller one from its corresponding array.
4. Repeat Step 3 until both arrays are exhausted. If $S = \phi$, stop—there are no direct connecting lines. Otherwise, go to Step 5 in the next phase.

Algorithm A1: Direction Check Phase

5. For each element in *S*, check if Stop *A* is upstream of Stop *B*. If not, remove it from *S*; otherwise, continue.
6. The solution is found as *S*. If $S = \phi$, there are no direct lines connecting from *A* to *B*.

Algorithm *A*1 is simple and efficient. The computational time of the sort operation in Step 2 is to the order of $L(\log L)$, where *L* is the number of overlapping bus lines passing a bus stop. In practice, the set of bus lines passing *A* or *B* can be sorted and filed before the on-line implementation of the algorithm. Therefore, the algorithm *A*1 can be easily implemented as a linear-time algorithm with order of $O(L)$, and it can serve as an efficient building block for other related search algorithms.

CONNECTION TRIPS WITH ONE TRANSIT TRANSFER

When no direct lines connect two bus stops, transit transfer activities are inevitable. For simplicity, connection trips with two or more transfers are not considered in this paper. We are concerned with finding all possible connection lines, with one transit transfer between two bus stops. This is more difficult than finding direct connecting lines, mainly because the transfer stop is not necessarily unique and the multiple transfer stops are hard to identify. A scenario of a connecting trip with seven possible transfer stops is shown in Figure 1.

Finding the transfer points between two stops *A* and *B* is complex because every downstream stop of *A* along any bus line passing *A* as well as every upstream stop of *B* along any bus line passing *B* is a potential transfer point and must be traced and checked. Figure 2 illustrates such a situation, in which every one of the stops of a_is and b_js $i, j = 1, 2, \ldots, 7$, is a potential transfer stop. Consequently if one uses the complete-enumeration type of pairwise comparison to search for the transfer stops, it takes approximately the order of L^2S^2 iterations simply to find the potential transfer stops, where *L* is the number of overlapping bus lines passing one stop and *S* is the average number of bus stops per line.

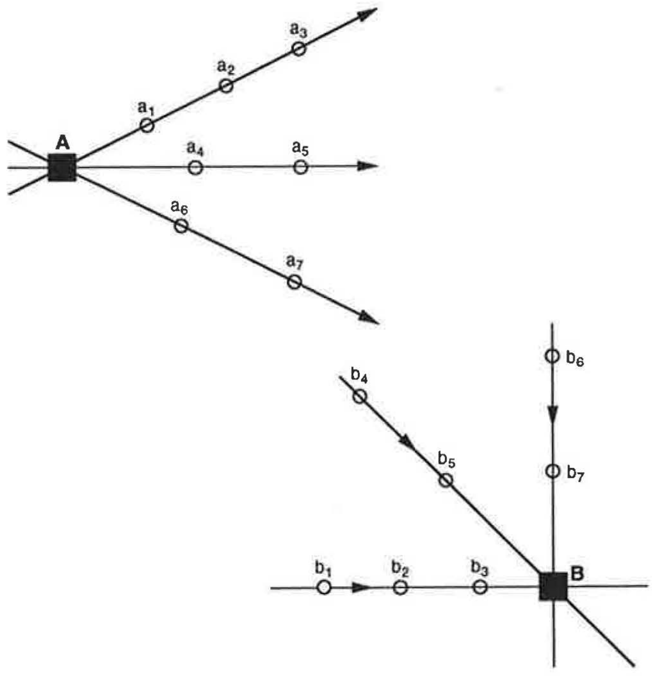


FIGURE 2 Potential transfer points of connecting trip.

The aforementioned complete-enumeration algorithm is a polynomial-time algorithm of $O(L^2S^2)$. It is still efficient in terms of combinational optimization. However, for practical on-line information service applications, the response time of such an algorithm may be too slow to accept. According to our implementation experience in Taipei, this algorithm when implemented by a PC 286/16 AT microcomputer took 20 to 90 sec to find the routing information for connecting trips. Such a performance was considered unacceptable because nobody is willing to hold the phone and wait for 20 sec or longer. This also led to the development of a much more efficient search algorithm.

SEARCH ALGORITHM USING HASH-TABLE TECHNIQUES

The full routing information of connection trips from A to B with one transit transfer can be considered as path information represented as $A-X-B$, where X is the transfer point. However, from a microcomputer it is difficult to find such a piece of full information at once. Our solution is to decompose the problem into subproblems and then combine the partial information obtained to form the final solution. Generally speaking, four subproblems are involved here:

- ($S1$)—Identification of all possible transfer points, the Xs;
- ($S2$)—Identification of bus lines connecting from A to B, the $X-B$ part of routing information;
- ($S3$)—Identification of bus lines connection A to X, the $A-X$ part of routing information; and
- ($S4$)—Combination of $A-X$ and $X-B$ to form the full information of $A-X-B$.

The subproblems can be solved efficiently by the use of hash-table, or hashing, techniques. Hashing is one of the data structure techniques commonly used to handle symbol tables in computer science (*16*). Considering the limited RAM storage space on a microcomputer, we used a single hash table of 32 × 32 bytes to keep track of all the potential transfer stops. The hash table is partitioned into 32 buckets (in bytes) and each bucket is capable of holding 256 records (in bits). A hashing function $f(X)$ maps the identifier X—that is, the code of a bus stop—to the address of X in the hash table. The typical code of a bus stop is $X = NNyyy$, where NN is the zonal number, $NN = 1, 2, \ldots, 32$, and yyy is the stop number, $yyy = 1, 2, \ldots, 256$. We defined the following hashing function to set up the hash table used in our algorithm:

$$f(NNyyy) = (NN - 1) \times 256 + yyy \qquad (1)$$

The hash table can be considered as 32 × 256 matrix. Each element in the matrix is a binary digit that indicates the unique address of its corresponding bus stop. The address is well defined by the hashing function of Equation 1.

Our Algorithm A2, designed for the search of full routing information of $A-X-B$, has five major steps. The framework of the algorithm design is shown in Figure 3. A detailed description of the algorithm A2 (for connecting transit trips) is given as follows:

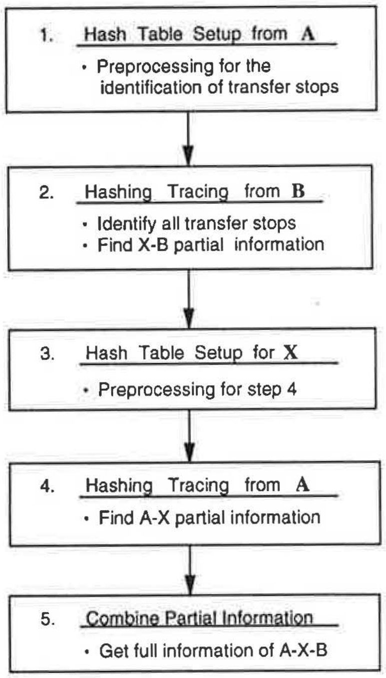

1. Hash Table Setup from A
• Preprocessing for the identification of transfer stops
2. Hashing Tracing from B
• Identify all transfer stops
• Find X-B partial information
3. Hash Table Setup for X
• Preprocessing for step 4
4. Hashing Tracing from A
• Find A-X partial information
5. Combine Partial Information
• Get full information of A-X-B

FIGURE 3 Design framework of Algorithm A2.

1. Use Equation 1 to define the hash table for all the downstream stops of A along all the bus lines passing A. All the corresponding addresses of the bus stops such as $a_1, a_2, \ldots, a_7$ (Figure 2) in the hash table are turned on.
2. Select any one line that passes B and for every upstream bus stop of B along that bus line, use the hashing function to map the stop to its corresponding address in the hash table, and check if the address is on. If the corresponding address is on, it is a transfer stop, X; otherwise, continue to check the next bus stop.

Repeat the previous step until all bus lines passing B are traced. Now we have obtained the set of all possible transfer stops and the corresponding bus lines leading to the destination stop B, that is, the routing information about the $X-B$ part.

3. Clear the hash table developed in Step 1, and use the hashing function (Equation 1) to rebuild a hash table for all the transfer stops found in Step 2.
4. Pick one of the bus lines passing stop A, and for every downstream bus stop of A along that line, use the hashing function to check if its corresponding address in the new hash table is on. If yes, keep the record of the transfer stop X as well as the line connecting A to X; otherwise, continue to check the next stop.

Repeat the previous step until all bus lines passing A are traced. Now we have obtained the routing information about the $A-X$ part.

5. Combine partial information of $A-X$ and $X-B$ by using the sort-and-compare techniques similar to those described in Algorithm A1: sort the two ways of Xs in ascending order and do pairwise comparison to combine $A-X$ and $X-B$ for every common transfer stop.

6. Postprocess the route information to rank the alternative connecting paths in preference order based on shortest path or other criteria such as maximal transfer combinations (optional).

Algorithm A2 is much more efficient than the complete-enumeration type of algorithm mentioned earlier. Without using hashing functions, the aforementioned algorithm requires $O(L^2S^2)$ time just to solve the subproblem $S1$. And by using hash-table structures, the computational time of the algorithm for solving the whole problem can be reduced to the order on LS. It is because in each of Steps 1 through 4 of the algorithm, the setup and tracing of a hash table requires only a constant amount of computational time for each bus stop processed, and the number of all potential transfer stops is $O(LS)$. Step 5 involves sort-and-compare procedures, thus it requires time comparable to the order of $S_x[\log(S_x)]$, where S_x is the number of transfer stops of the connecting trip A–X–B. For practical applications in transit systems, the number of all potential transfer stops, LS, usually is much greater than $S_x[\log(S_x)]$. Therefore, the computational time of Algorithm A2 is $O(LS)$.

Note that the $O(LS)$ algorithm is much more efficient than the $O(L^2S^2)$ algorithm described in the previous section. For the Taipei Transit System, the order of LS ranges from 10 to 10^3. This explains why the on-line implementation of Algorithm A2 can run hundreds of times faster than that of the complete-enumeration algorithm and reduce the computational time from minutes to seconds.

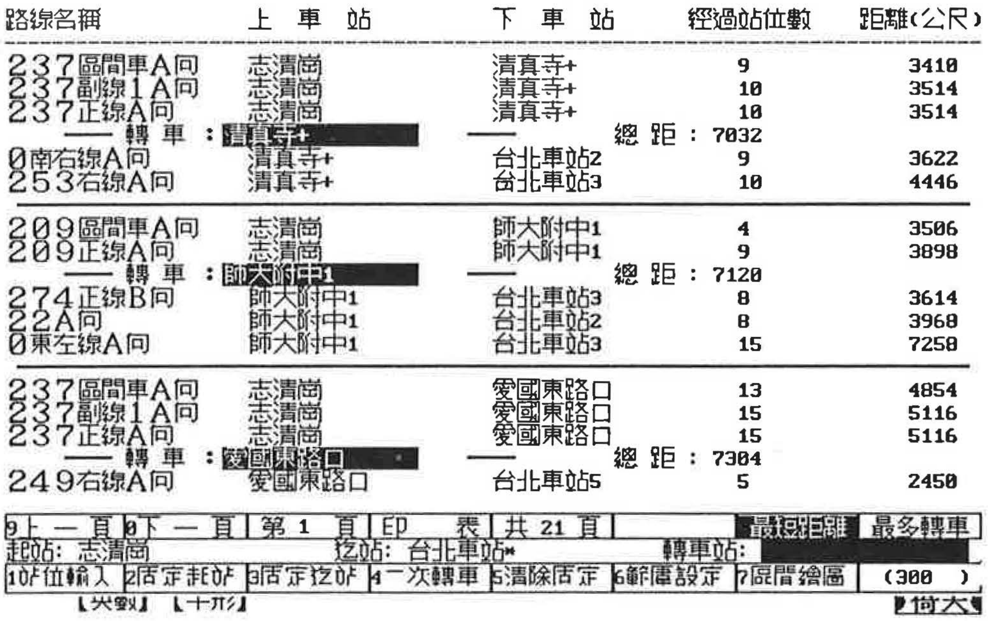
Algorithm A2 is primarily designed for microcomputer applications. The hash table takes only about 1K, that is, 32 × 32 = 1,024 bytes, of RAM space, making it very efficient for applications in MS-DOS. With other computing facilities, one can use hash tables bigger than 32 × 32 bytes to store the data of both transfer stops and connecting lines, and to reduce the number of steps as proposed in our algorithm. However, the use of a bigger hash table may not improve the implementation time of the algorithm.

IMPLEMENTATION RESULTS AND CONCLUSIONS

Algorithms A1 and A2 have been successfully running on a microcomputer-based transit information system in Taipei since January 1991. Taipei is the capital city of Taiwan, the Republic of China (ROC). The city of Taipei has an area of 272 km^2 within its administrative boundaries and a population of about 2.7 million. Bus transit services in the Taipei city are primarily provided by 10 major bus companies that have joined to form the United Operating Center (UOC) of the Taipei Transit System. As of August 1991, the UOC operates more than 250 routes with 3,174 buses, carrying approximately 2.1 million passenger trips a day (*17*).

The transit routing information service system in Taipei is a telephone on-line service system. Two shifts of trained operators answer the telephone inquiries with the help of computerized route information system. The computerized system is programmed in Turbo C and implemented on two IBM PC 386/20 AT microcomputers with math coprocessors and other peripherals. Specifically, the system operates with two telephone lines, (02)321-2000 and (02)341-2000, and two independent computerized route information systems in order to provide maximal on-line services to the public. On average, the system receives and answers about 105 calls each day.

Because of the complicated transit network structure, a significant portion of transit trips in Taipei involve transfers. It was estimated that approximately 1 million passenger trips a day are made through bus transfers (*13*). Therefore, most callers ask for routing information of transfer or connecting trips. The system would first check if there were direct connecting lines. If not, it would automatically search for the bus line connecting through transfer stops and list on the monitor screen the connecting trips in order of shortest distance or maximal transfer combinations, depending on the choice of the rider.



```
路線名稱          上  車  站           下  車  站      經過站位數     距離(公尺)
---------------------------------------------------------------------------
237區間車A向      志清崗               清真寺+            9             3410
237副線1A向       志清崗               清真寺+           10             3514
237正線A向        志清崗               清真寺+           10             3514
   —— 轉  車  : 清真寺+              ——      總 距 : 7032
0南右線A向        清真寺+              台北車站2          9             3622
253右線A向        清真寺+              台北車站3         10             4446
---------------------------------------------------------------------------
209區間車A向      志清崗               師大附中1          4             3506
209正線A向        志清崗               師大附中1          9             3898
   —— 轉  車  : 師大附中1            ——      總 距 : 7120
274正線B向        師大附中1            台北車站3          8             3614
22A向             師大附中1            台北車站2          8             3968
0東左線A向        師大附中1            台北車站3         15             7258
---------------------------------------------------------------------------
237區間車A向      志清崗               愛國東路口        13             4854
237副線1A向       志清崗               愛國東路口        15             5116
237正線A向        志清崗               愛國東路口        15             5116
   —— 轉  車  : 愛國東路口           ——      總 距 : 7304
249右線A向        愛國東路口           台北車站5          5             2450

|9上 一 頁|0下 一 頁| 第 1  頁| 印    表| 共 21 頁|          |最短距離|最多轉車|
起站: 志清崗                迄站: 台北車站*               轉車站:
|1站位輸入|2固定起站|3固定迄站|4一次轉車|5清除固定|6範圍設定|7區間繪圖| (300  ) |
     【英數】 【半形】                                                  【倚天】
```

FIGURE 4 Typical screen output of connecting trip information in Taipei Transit System.

A scenario of a connecting trip with seven possible transfer stops was shown earlier in Figure 1. For the dense transit network operated by 10 bus companies in Taipei, it is not uncommon to find connecting trips with more than 10 transfer stops. The full routing information of $A-X-B$ thus can be as long as 20 to 30 (screen) pages. Figure 4 shows a hard copy of the first-page screen output of a 21-page full output file. The routing information listed on the top of the screen tells a person aboard at Chih-Chin Ridge can take Line 237 (zonal, express, or regular) to Mosque and make a transfer from there to Line 0–south or Line 253R to get to the destination stop, the Taipei Train Station. This is the shortest-distance path of the person's total of 38 connecting path alternatives. Algorithm A2 works fine for the system. Taking only 2 or 3 sec to find the 21-page output file and show the first-page output on the monitor screen.

The algorithms proposed in this paper should be useful for microcomputer-based routing information systems for an integrated public transportation system. The techniques of using a small hash table to implement the search algorithm efficiently on a microcomputer appear to be useful for other microcomputer applications as well. With modifications or extensions, Algorithm A2 might be applied to routing information systems for airlines or the track-and-trace systems for courier service companies. Such potential applications need further investigation.

ACKNOWLEDGMENT

This research was jointly supported by the National Science Council, ROC and the Taipei Transit System UOC. An earlier preliminary study of this research was supported by the Institute of Transportation (IOT), Ministry of Transportation and Communications, ROC. The authors are grateful to Chia-Juch Chang, Director General of IOT, for his initiative support.

REFERENCES

1. R. G. P. Tebb. Travel Information Research at TRRL. Presented at the 11th Annual Seminar on Public Transport Operations Research, University of Leeds, England, July 1979.
2. C. O. Tong and A. J. Richardson. Computer General Travel Information for Urban Transit Networks. *Proc., 10th Australia Transport Research Forum*, Melbourne, Vol. 2, May 1985, pp. 197–213.
3. C. J. Vanigrok. *Designing an Integrated Travel Information System*. Monograph, Instituut TNO Wiskunde Infomatieverwerking en Statistiek, Delft, the Netherlands, Sept. 1982.
4. M. W. Pickett. The Production, Dissemination and Costs of an Integrated Public Transport Travel Information System. Report SR657. U.K. Transport and Road Research Laboratory, Crowthorne, Berkshire, England, 1981.
5. J. J. Fruin. *NCTRP Synthesis of Transit Practice 7: Passenger Information Systems for Transit Transfer Facilities*. TRB, National Research Council, Washington, D.C., 1985.
6. K. G. Baass, B. Allard, and R. Chapleau. Traffic and Transport Data Management by Micro-Computer. *Proc., 2nd North American Conference on Microcomputer Applications in Transportation*, ASCE, 1987, pp. 536–545.
7. D. Knight and B. Albee. SCT's Experience with Automated Fleet Management. *Proc., 2nd North American Conference on Microcomputer Applications in Transportation*, ASCE, 1987, pp. 474–483.
8. J. McCarthy. Using Microcomputers for Maintenance Management in a Small Transit Agency. *Proc., 2nd North American Conference on Microcomputer Applications in Transportation*, ASCE, 1987, pp. 387–394.
9. D. J. Wahl. Application of Data Processing Techniques to the Monitoring of Transit System Performance in San Diego. In *Transportation Research Record 1050*, TRB, National Research Council, Washington, D.C., 1985, pp. 53–56.
10. J. M. Ward and M. J. Demetsky. Computerization of Transit Performance Evaluation. *Proc., 2nd North American Conference on Microcomputer Applications in Transportation*, ASCE, 1987, pp. 336–347.
11. M. R. Cutler. Impact of Technology and Labor Management Strategies on the Efficiency of Telephone Information Services. In *Transportation Research Record 1039*, TRB, National Research Council, Washington, D.C., 1985, pp. 1–9.
12. A. Han and N. Wilson. The Allocation of Buses in Heavily Utilized Networks with Overlapping Routes. *Transportation Research*, Vol. 16B, 1982, pp. 221–232.
13. A. F. Han. Assessment of Transfer Penalty to Bus Riders in Taipei: A Disaggregate Demand Modeling Approach. In *Transportation Research Record 1139*, TRB, National Research Council, Washington, D.C., 1988, pp. 8–14.
14. R. Kwan. Coordination of Joint Headway. In *Computer-Aided Transit Scheduling* (J. Daduna and A. Wren, eds.). Springer-Verlag, New York, N.Y., 1988, pp. 304–314.
15. A. Han and H. L. Chang. *Route Structure Analysis of the Taipei Transit Network* (in Chinese). Research report. Department of Transportation Engineering and Management, National Chiao Tung University, Taiwan, 1987.
16. E. Horowitz and S. Sahni. Hashing. In *Fundamentals of Data Structure in PASCAL*, 3rd ed. Computer Science Press, New York, N.Y., 1982, Chapter 9.
17. *The Statistical Abstract of Taipei Traffic* (in Chinese). Taipei Municipal Government Office, Taiwan, ROC, Nov. 1991.

Publication of this paper sponsored by Committee on Transportation Supply Analysis.