Artificial Intelligence–Based System Representation and Search Procedures for Transit Route Network Design

M. HADI BAAJ AND HANI S. MAHMASSANI

An artificial intelligence (AI)–based representation of transportation networks is described. Such representation facilitates the development of efficient AI search algorithms that make up the bulk of the computational effort involved in the design and analysis of transportation networks. The novel representation is demonstrated, as are its advantages as implemented in the AI search algorithms developed for the design and analysis of a particular type of transportation network, namely, transit bus routes networks.

The purpose of this paper is to describe an artificial intelligence (AI)–based representation of transportation networks. Such representation facilitates the development of efficient AI search algorithms that make up the bulk of the computational effort involved in the design and analysis of transportation networks. We demonstrate the novel representation and its advantages as implemented in the AI search algorithms developed for the design and analysis of a particular type of transportation network, namely, transit bus route networks. From an implementation perspective, using AI search techniques offers the advantage of representing the transit network design problem (TNDP) and carrying out a search efficiently using the "list" data structure of Lisp (List Programming), a so-called fifth-generation computer language (*1*).

TNDP

Several authors have studied the TNDP (*2*). In the TNDP, one seeks to determine a configuration, consisting of a set of transit routes and associated frequencies, that achieves some desired objective, subject to the constraints of the problem. Mathematical formulations of the TNDP have been concerned primarily with minimizing an overall cost measure, generally a combination of user costs and operator costs. The former is often captured by the total travel time incurred by users in the network, whereas a proxy for operator costs is the total number of buses required for a particular configuration. Feasibility constraints may include, but are not limited to, (a) minimum operating frequencies on all or selected routes (policy headways, where applicable), (b) a maximum load factor on any bus route, and (c) a maximum allowable bus fleet size.

M. H. Baaj, Department of Civil Engineering, Arizona State University, Tempe, Ariz. 85287. H. S. Mahmassani, Department of Civil Engineering, University of Texas, Austin, Tex. 78712.

Most existing formulations can be viewed as variants of the following mathematical program:

Minimize

$$\left\{c_1\left[\sum_{j=1}^{n}\sum_{i=1}^{n} d_{ij}t_{ij}\right] + c_2\left[\sum_{\text{all } k \in \text{SR}} f_k T_k\right]\right\} \tag{1}$$

Subject to

$$\text{Frequency feasibility: } f_k \geq f_{\min} \qquad \text{for all } k \in \text{SR} \tag{2}$$

$$\text{Load factor constraint: } \text{LF}_k = \frac{(Q_k)_{\max}}{f_k \text{CAP}} \leq \text{LF}_{\max}$$

$$\text{for all } k \in \text{SR} \tag{3}$$

$$\text{Fleet size constraint: } \sum_{\text{all } k \in \text{SR}} N_k = \left[\sum_{\text{all } k \in \text{SR}} f_k T_k\right]$$

$$\leq W \qquad \text{for all } k \in \text{SR} \tag{4}$$

where

d_{ij} = demand between nodes i and j;
t_{ij} = total travel time between i and j = $t_{\text{invtt},ij} + t_{\text{wt},ij} + t_{\text{tt},ij}$;
$t_{\text{invtt},ij}$ = in-vehicle travel time between nodes i and j;
$t_{\text{wt},ij}$ = waiting time incurred while traveling between nodes i and j;
$t_{\text{tt},ij}$ = transfer time incurred while traveling between nodes i and j;
N_k = number of buses operating on route k; $N_k = f_k T_k$;
f_k = frequency of buses operating on route k;
$f_{\min}$ = minimum frequency of buses operating on any route;
T_k = round trip time of route k;
W = fleet size available for operation on the route network;
LF_k = load factor of route k;
$(Q_k)_{\max}$ = maximum flow occurring on any link of route k;
CAP = seating capacity of buses operating on the network's routes;
SR = set of transit routes; and
c_1, c_2 = weights reflecting the relative importance of the two cost components.

Our proposed solution approach is hybrid in nature in that it provides a framework to incorporate the knowledge and expertise of transit network planners, efficient search techniques using AI tools, and some algorithmic procedures developed by others or adapted from related problems in vehicle routing. The three major components in the proposed approach are a route generation design algorithm (RGA) that generates different sets of routes corresponding to different trade-offs among the principal objectives; an analysis procedure (TRUST) that computes an array of network-, route-, and node-level descriptors as well as the frequencies of buses necessary on all routes to maintain load factors under a prespecified maximum; and a route improvement algorithm (RIA) that considers each set of routes and uses the result of the analysis procedure to generate an improved set of routes (*3*).

The RGA is a design algorithm that configures, for a given set of nodes connected by a road network and demand matrix, sets of routes that correspond to different trade-offs between the user and operator costs. It queries the user for the minimum percentage of the total demand that is to be satisfied directly (i.e., without transfers) and the percentage of the total demand that is to be satisfied with no more than two transfers. It searches the demand matrix for high-demand node pairs and selects them as seeds for the initial set of skeletons. These skeletons are expanded to routes by way of different node selection and insertion strategies.

The knowledge and expertise of transit planners is implemented in the different routines in the form of constraints on search and within the different node selection and insertion strategies. Different targets for the demand satisfaction and different insertion strategies result in different sets of routes with different user and operator costs. RGA relies on algorithmic procedures such as the *k*-shortest-paths algorithm (*4*) and on the selective application of the transit planners' knowledge and expertise to guide the search.

Once sets of routes are generated, an analysis procedure called TRUST (Transit Routes Analyst) is called to evaluate those alternative transit network route configurations. TRUST computes a variety of performance measures reflecting the quality of service and costs experienced by the users and the resources required by the operator. An essential feature of TRUST is the computation of service quality measures in terms of the fraction of trips with different number of transfers. Also important are the summary measures of transfer activity by route and by node.

After RGA has generated and TRUST has evaluated the sets of routes, the RIA is called to improve each of the generated sets. These modifications can be classified into two groups of actions: actions on the transit system coverage level, and actions on the route structure level. The first goal that RIA was designed to achieve is that of making the sets of routes generated by RGA economically and operationally feasible. RIA considers two modifications: discontinuing these low ridership routes and joining these routes or their nodes with other medium- to high-ridership routes. The second goal of RIA is to demonstrate and test existing improvement procedures. The modifications that RIA considers are route splitting and branch exchange heuristics (whereby branches of different routes are exchanged to form new routes so as to reduce transfers at the intersection nodes).

AI-BASED TRANSIT NETWORK REPRESENTATION

According to Rich and Knight (*5*), a good system representing knowledge in a particular domain should possess the following properties:

1. Representational adequacy: the ability to represent all kinds of knowledge that are needed in that domain.
2. Inferential adequacy: the ability to manipulate the representational structures in such a way as to derive new structures corresponding to new knowledge inferred from old.
3. Inferential efficiency: the ability to incorporate into the knowledge structure additional information that can be used to focus the attention of the inference mechanisms in the most promising directions.
4. Acquisitional efficiency: the ability to acquire new information easily. The simplest case involves direct insertion, by a person, of new knowledge into the data base.

Our Lisp-based representation of the TNDP offers advantages over conventional languages such as FORTRAN, C, or Pascal in terms of knowledge representation and search processing. We implement an object-oriented hierarchical structural representation: nodes are connected by links, which in turn are traversed by routes. The routes combine to form the paths by which a node pair's demand is assigned. The set of routes and their associated bus frequencies define the transit network object.

The transit network data representation lends itself conveniently to the "list" data structure representation of Lisp, which in turn supports the kind of path search strategies of interest in this application. This can be illustrated by the following:

- The network connectivity can be conveniently represented in a descriptive language such as Lisp: to each network node, one associates a set (or, in Lisp, a list) of neighboring nodes as well as the trip time (cost) associated with the nodes. Thus, the list {2[(1 11.4)(3 2.9)(6 8.0)]} indicates that one can travel from Node 2 to Node 1 in 11.4 min, to Node 3 in 2.9 min, and to Node 6 in 8 min. In addition, with each node object one associates properties whose values are useful to the design and analysis procedures. Such properties include the demand originating at a given node (and the percentage assigned by the network under design), the demand destined to a given node, and the number of trips transferring at a given node (all indicators of the node's relative importance).
- A route can be represented as a list of nodes, thus Route r25 is defined by the list of nodes (18 11 10 9 8 12 14). Associated with the route object are the list of flows on the routes links and the number of buses deployed to maintain the load factor below a minimum value that is prespecified by the user.
- The search techniques that are specific to the TNDP can be readily programmed in Lisp. In such techniques, a feasible path connecting two network nodes can be represented as a list. Thus, the list [(r1 9 16)(r8 16 21)] implies that one can travel from Node 9 to Node 21 by boarding Route r1 from Node 9 to Node 16 and Route r8 from Node 16 to Node 21 (i.e., Node 16 is a transfer node).

DESIGN PHILOSOPHY OF AI SEARCH ALGORITHMS

Figure 1 presents the principal computational trade-offs that should be considered in the design of AI search techniques. As Nilsson argued, the computational costs of any AI search algorithm can be separated into two major components: the rule application costs and the control costs (*6*). A completely uninformed control system is characterized by a small control strategy cost because arbitrary rule selection need not depend on costly computations. However, such a strategy results in high rule application costs because generally it must try a large number of rules to find a solution. To inform a control system completely about the problem typically involves a high-cost control strategy, in terms of the storage and computations required. However, such a completely informed control strategy results in minimal rule application costs, for they guide the search directly to a solution.

The efficiency of an AI search technique is thus directly tied to achieving a proper balance between both computational cost components. This relies on "informedness," or the amount of knowledge and information that the rule-selecting computations possess about the problem at hand. Optimum search efficiency is usually obtained from control strategies that are less than completely informed. Thus, all three major components of our AI-based solution approach constitute a testing ground for selective application of knowledge.

EFFICIENCY OF AI-BASED DESIGN AND ANALYSIS SEARCH PROCEDURES

The principal motivation for using Lisp (or, more generally, a fifth-generation language) lies in the nature of the computational activity taking place in our solution approach, which consists of searching and screening paths in a graph. It has been common wisdom in transportation network applications to avoid any form of path enumeration. Thus, most existing

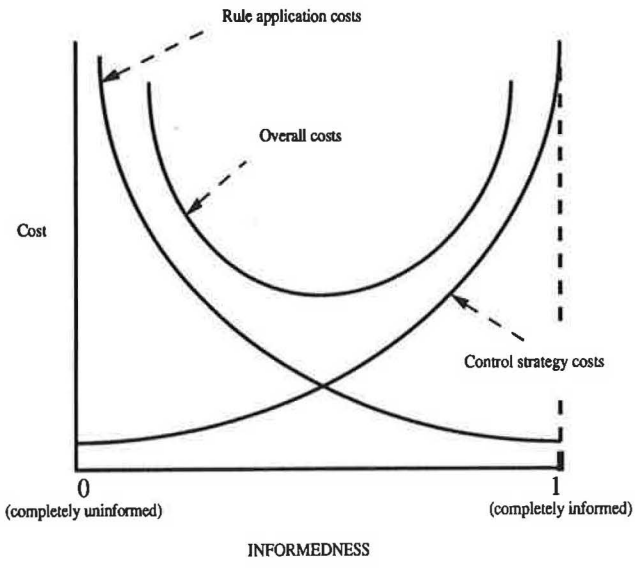

FIGURE 1 Computational costs of AI production system.

assignment procedures are limited to shortest-path constructs. However, other programming paradigms and advances in computing hardware and software can greatly facilitate some degree of path search and enumeration, which is justified by the added realism that it could allow into the resulting procedure.

Taylor describes simple programs written in Prolog, another fifth-generation language, to solve different route selection problems (*7*). His examples underscore the brevity of code as well as the relative ease of programming with fifth-generation languages. At the basis of these programs are some general "predicates" (Prolog meta-statements) that test for set membership, generate the intersection or union of any two lists as well as the complement of one list in another, sort a list of objects according to some numerical property, or append a new element to a set. Such meta-statements define the necessary condition for the required solution, thus relieving the program developer from worrying about the elemental computing and housekeeping chores, as would be the case with conventional programming languages such as FORTRAN, C, and Pascal.

An example of the use of such predicates in one AI algorithm is the way TRUST assigns a given node pair's demand to the transit network generated by RGA. For a given node pair (i,j), the list of routes passing through node i and the list of routes passing through node j (denoted by SR1 and SR2, respectively) are assembled by calling the "routes-passing-by" procedure for both nodes. If either of the lists are empty, then at least one of the two nodes (i or j) is not served by any transit route, and the demand d_{ij} cannot be assigned. If both lists are not empty, than a call is made to procedure "assign-0-transfer?," which checks whether the demand can be assigned directly (i.e., without transfers). This is possible only if the intersection list (of the two lists SR1 and SR2), which is the subset of all routes that have both nodes i and j on their node list, is not empty. When this is the case, the intersection list is passed on to the procedure "decide-0," whose function is to distribute the demand d_{ij} among the acceptable routes.

If the "assign-0-transfer?" procedure is unable to assign the demand between i and j directly (with no transfers), a call to procedure "assign-1-transfer?" is made. The latter checks whether the trip can be completed with one transfer. This check is carried out by examining, for every possible combination of a route member of List SR1 (say, R1) and another of List SR2 (say, R2), the intersection list of the list of nodes of R1 and R2. If the intersection list is not empty, then its contents are possible transfer nodes between R1 and R2. For example, if the intersection list is (tf1 tf2), TRUST forms two possible paths for the assignment of demand between nodes i and j: [(R1i tf1)(R2 tf1 j)] and [(R1 i tf2)(R2 tf2 j)]. The first indicates that a possible path from i to j consists of boarding Route R1's bus at i and staying on it until tf1 (Transfer Node 1) is reached. There the passenger should transfer to Route R2 and travel on it until the destination node j. For each possible path involving one transfer, an estimate of the total travel time is calculated. All paths between i and j whose total travel times exceed the minimum possible value by more than a specified threshold (say, 10 percent, as selected here) are rejected. Procedure "decide-1" subsequently distributes d_{ij} among the paths that have passed the filtering process.

Similarly, if it is determined that d_{ij} cannot be assigned with at most one transfer, procedure "assign-2-transfer?" is called to search for paths involving two transfers. If no such path is found, the demand between i and j will remain unsatisfied. In other words, it is assumed that a passenger will simply not consider boarding the transit buses to accomplish a trip that requires three or more transfers. Hence, TRUST avoids searching for paths that reach destination j with three or more transfers, thereby avoiding an otherwise considerable amount of meaningless search and keeping the execution time within tolerable limits.

The "assign-2-transfer?" procedure searches for all paths with exactly two transfers between given nodes i and j. The search consists of finding a route that passes through neither i nor j but that shares a node with a route passing through i (i.e., with a member of SR1) and another through node j (i.e., with a member of SR2). List SR3 of routes that pass through neither i nor j is obtained as the complement (in SR, the set of all routes) of the union list of the previously generated lists, SR1 and SR2. For a trip to require exactly two transfers between origin i and destination j, the first route, R1, must pass by node i (hence, R1 $\in$ SR1); the second route, R3, must pass by some node other than i or j (hence, R3 $\in$ SR3); and the third route, R2, must pass by node j (hence, R2 $\in$ SR2). Thus, three DO loops are executed: the outer one on SR3, the inner one on SR1, and the innermost one on SR2. A route from each of the above three sets is selected, and the following test is performed: if the "list-of-nodes" of the route from SR3 (say, R3) intersects both the "list-of-nodes" of the route from SR1 (say, R1), and the "list-of-nodes" of the route from SR2 (say, R2), then a possible two-transfer path is defined. For example, if the intersection of the "list-of-nodes" of R3 and R1 is (tf1 tf2) and that of the "list-of-nodes" of R3 and R2 is (tf3), then TRUST defines two possible paths: [(R1 i tf1)(R3 tf1 tf3)(R2 tf3 j)] and [(R1 i tf2)(R3 tf2 tf3)(R2 tf3 j)]. This is repeated until all possible combinations of triplets (R3 $\in$ SR3, R1 $\in$ SR1, R2 $\in$ SR2) are checked and listed.

On the negative side, Lisp, like most higher-level languages, may experience relatively slower computational performance when it comes to mathematical computations (as opposed to symbolic manipulations). However, in our particular application, the tests conducted and reported by Baaj indicate that the AI search algorithms perform satisfactorily within reasonable execution times (*3*).

CONCLUSIONS

In this paper we described an AI-based representation of transportation networks. Such representation facilitated the development of efficient AI search algorithms that composed the bulk of the computational effort involved in the design and analysis of transportation networks. AI search techniques offer the advantage of representing the TNDP and carrying out search efficiently using the list data structure of Lisp. Such representation was essential in the success of our AI-based hybrid solution approach proposed for the solution of the TNDP. Our hybrid approach consisted of (a) AI heuristics for transit route generation and improvement, (b) a transit network evaluation model, and (c) the systematic use of context-specific knowledge to guide the search techniques. Results of computational testing of our AI-based solution approach on a benchmark transit network and on data generated for the transit network of the city of Austin, Texas, were promising (*3*). Further testing remains to be done on different transit networks and their corresponding transit demand matrices. In addition, we seek to investigate the merits of applying an AI-based representation and solution approach in other transportation network design problems.

REFERENCES

1. P. H. Winston and B. K. P. Horn. *Lisp*, 3rd ed. Addison-Wesley Publishing Company, Inc., Reading, Mass., 1989.
2. M. H. Baaj and H. S. Mahmassani. TRUST: A Lisp Program for the Analysis of Transit Route Configurations. In *Transportation Research Record 1283*, TRB, National Research Council, Washington, D.C., 1990, pp. 125–135.
3. M. H. Baaj. *The Transit Network Design Problem: An AI-Based Approach*. Ph.D. thesis. University of Texas, Austin, 1990.
4. D. R. Shier. On Algorithms for Finding the K Shortest Paths in a Network. *Networks*, Vol. 9, 1979, pp. 195–214.
5. E. Rich and K. Knight. *Artificial Intelligence*. McGraw-Hill, Inc., New York, N.Y., 1991.
6. N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing Company, Palo Alto, Calif., 1980.
7. M. P. A. Taylor. *Knowledge-Based Systems for Transport Network Analysis: A Fifth-Generation Perspective on Transport Network Problems*. Department of Civil Engineering, Monash University, Victoria, Australia, 1989.

Publication of this paper sponsored by Committee on Expert Systems.