Transportation Network Investment Problem—A Synthesis of Tree-Search Algorithms

YUPO CHAN

ABSTRACT

In this paper tree-search algorithms that are particularly adept at solving network-design problems in transportation planning are surveyed and synthesized. A unified view of the underlying principles of these tree-search algorithms is presented. Two methodologies--branch-and-bound and branch-and-backtrack--have been identified as promising techniques for solving typically nonlinear and ill-behaved network-design problems, particularly when they are coordinated with the postoptimality procedures of link lengthening and link shortening in minimum-path computation. The two algorithms are then compared, and a third algorithm--based on double bounding--is synthesized to solve transportation network-design problems more efficiently.

A number of problems in transportation planning deal with network investment or network design. An example may be the improvement of a rail or highway network where the heavy capital investment involved necessitates a careful configuration. A body of literature exists on this type of analysis, which is often referred to as the link-addition problem. This paper is written to summarize the pertinent techniques that address the problem.

The plan of presentation is as follows: First, the essential elements of the mathematical formulation of a transportation network-design problem are identified. Second, a brief review of the solution methods, which lead to the potential of the tree-search technique, is presented. Third, examples of the upper- and lower-bound tree-search techniques are given, compared, and their key features uncovered. The comparison helps to arrive at a generalized bounding technique to solve network-design problems.

PROBLEM STATEMENT

The substantive problem of this paper can be stated as follows: The transportation planner is given a fixed budget, B, to improve a multiple origin-destination network. Each link in the network is associated with a level-of-service function $C_{ij}(X_{ij})$, which is a monotonically increasing function of flow, X_{ij}. Investment projects are defined for a link (i,j), where $\Delta C_{ij}(X_{ij})$ denotes the improvement on link (i,j). It is assumed that the project candidates have been identified (i.e., ΔC_{ij}'s are exogenously defined for a subset of the links). The problem is remotely similar to a knapsack problem in the sense that an attempt is made to fit a number of projects, each with a nonzero cost of b_{ij}, into the budget:

$$\sum_{(ij)} b_{ij}\,\delta_{ij} \leq B \tag{1}$$

Department of Civil and Environmental Engineering, Washington State University, Pullman, Wash. 99164.

where δ_{ij} is a 0-1 variable that denotes whether project ΔC_{ij} is rejected (0) or accepted (1). A shorthand form for Equation 1 is $\underline{b}^T\underline{\delta} \leq B$.

This is a multicommodity network flow (1). Each "commodity" is defined as the vehicles, passengers, or cargo that start from an origin (O), k, heading for destination (D), ℓ. There are as many commodities as the number of origin-destination (O-D) pairs. The constraints can be written as a tableau of a block diagonal form; each block is a "copy" (2) of the node-arc incidence matrix, $A^{k\ell}$, representing the flow between k and ℓ. The flow between k and ℓ using link (i,j) is denoted by $X_{ij}^{k\ell}$'s, which are grouped into a vector $\underline{X}^{k\ell}$. Each copy $A^{k\ell}$ models an amount of flow $V^{k\ell}$ originating at k and terminates at ℓ. The node-arc incidence matrix $A^{k\ell}$ is composed of the following elements:

$$\sum_i X_{ip}^{k\ell} - \sum_j X_{pj}^{k\ell} = \begin{cases} -V^{k\ell} & \text{if } p = k \\ V^{k\ell} & \text{if } p = \ell \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where the O-D demand $V^{k\ell}$'s are functions of the level of service from k to ℓ [i.e., $V^{k\ell}(C^{k\ell})$]. If $\underline{d}^{k\ell}$ is used to denote the right-hand vector of Equation 2, each copy of commodity flow can be written as

$$A^{k\ell}\underline{X}^{k\ell} = \underline{d}^{k\ell}(\underline{C}^{k\ell}) \tag{3}$$

The flow on a link comes from diverse O-D pairs:

$$\sum_{k\ell \in R_{ij}} X_{ij}^{k\ell} = X_{ij} \tag{4}$$

where R_{ij} is the set of O-D flows that utilizes link ij.

The objective function minimizes the individual vehicle's travel cost (i.e., user optimizing instead of system optimizing) (3) for highway travel:

$$\text{Sup } Z = G\left\{\sum_{R^{k\ell}} \min \sum_{(ij)\in R^{k\ell}} [C_{ij}(X_{ij}) - \Delta C_{ij}(X_{ij})\,\delta_{ij}]\,X_{ij}^{k\ell}\right\} \tag{5}$$

where

$R^{k\ell}$ = the set of links contained in the routing from k to ℓ,
$G(\underline{\delta})$ = monotone function of the project vector $\underline{\delta}$, and
Sup(•) = max(•) or min(•).

For scheduled transportation services such as trains and airplanes, vehicular flow can be more appropriately modeled by a system-optimizing objective function (4,5), which is Equation 5 without the minimization operator between the summation signs. Notice that the amount of travel cost each project saves (henceforth called the "value" of a project) is not explicitly stated. The reason is that each time a project ΔC_{ij} is implemented, the flows X_{ij} may change because of a possible change in the minimum cost flow paths. The implementation of a particular project would affect the minimum cost flow pattern in a different way (and hence its value is different) depending on whether and what other projects have been implemented. The link-travel-time-reduction projects are termed dependent (6) because the value of a project depends on whether and what other projects have been implemented.

Obviously, variants of this "classic" formulation are found. Instead of minimizing the system or user cost, the total budget expenditure for a given level of effectiveness may be minimized (7). Furthermore, maximization of consumers' surplus and a system-equity measure (8) may be employed. Instead of a single period problem, a staged-investment formulation (8-11) can be used. Finally, a hierarchical approach to network investment (7,12) can be employed. The important point is that the tree-search method is flexible and robust enough to tackle all of these variants.

REVIEW OF SOLUTION METHODS

It is well recognized that there are serious limitations of the formal developments in mathematical programming for solving the typically ill-behaved transport network-design problem (13-15). Here a set of network-design methodologies, which combines the versatility of the enumerative-type algorithm with some analytical niceties of the algebraic formulations, is presented. These algorithms are referred to as tree-search solution algorithms in which the geometric configuration (the network synthesis problem) is structured by the enumerative mechanism and the passenger or commodity flow problem is solved by an algebraic formulation (the network analysis problem). In this way, a problem is decomposed into subproblems (4,16). The tree-search algorithms as defined here have the following additional advantages: First, the network-flow algorithms--such as traffic assignment--are computed only as needed and often involve postoptimality procedures in minimum-path computations. Second, the tree-pruning criteria are often stronger, thus delimiting the computational space. Finally (but probably most significant), the tree-search strategy as defined in this fashion indicates adaptability to the many more "real-life" issues encountered in transport network design--a point that will be elaborated.

TREE SEARCH

Included in the synthesis of tree search is a class of enumerative solution methods such as branch-and-bound and implicit enumeration (branch-and-backtrack) (7,8,17-30). Tree search derives its name in part from the way the solution procedure is graphically displayed as a directed tree (see example in Appendix). The solution strategy is to break up (or decompose) the original difficult problem into easier, auxiliary problems, each of which constitutes a network analysis problem.

An auxiliary problem is defined at the nodes of the directed tree. The first node of the tree is the root. In the directed tree, there is branching from a predecessor node to two or more successor nodes. Thus in the network-design problem cited previously $Z(\underline{\delta})$ is minimized or maximized by an optimal choice of $\underline{\delta}^*$. The solution strategy involves dividing the set of all feasible and infeasible solutions D into the combinatorial space of q subsets, where

$$D_1 \cup D_2 \cup \ldots \cup D_q = D \tag{6}$$

A partial solution is defined as one in which only a subset of the n decision variables has been assigned 0-1 values. Those decision variables not yet assigned a binary value are called the free variables. A completion of a partial solution is obtained by specifying binary values for the free variables.

In the directed tree, each auxiliary problem can be written as $Z_k(D_k)$ at node k. Among the current successor nodes, a lower bound can be computed for the corresponding network-analysis problem, yielding $Z_k^L(D_k)$ with

$$Z_k^L(D_k) \leq Z(\underline{\delta}) \qquad \underline{\delta} \in D \tag{7}$$

Such a bounded node is where branching takes place in the next step in the branch-and-bound algorithm. Likewise, an upper bound U is computed for the optimal solution $\underline{\delta}^*$.

A node is said to have been dominated if its objective function cannot be made better (than the objective function of a feasible solution already obtained) by further branching. Fathoming a node is the process of completing (explicitly or implicitly) the partial solution at that node. Inactive or fathomed nodes are nodes that have been considered and need not be investigated further because of dominance, feasibility, or end-of-branch considerations. In other words, if $Z_k^L > U$, the successor D_s cannot include the optimal solution $\underline{\delta}^*$. Hence these successors need not be examined further.

Active or unfathomed nodes, on the other hand, are nodes that still can be branched from. More precisely, if $Z_k^L \leq U$, the successor D_s may include $\underline{\delta}^*$. Active nodes that are not yet branched from are called terminal nodes.

Backtracking refers to "climbing up" the directed tree through the predecessor nodes to some terminal node and further branching from the terminal node. In the branch-and-bound procedure, typically, branching takes place from the best bound of all terminal nodes. In the branch-and-backtrack procedure, on the other hand, branching is done from the set of nodes that has been reached last (i.e., branch from the newest active node). Because all terminal nodes are considered (explicitly or implicitly) candidates for branching, this branching process is called flooding.

There are two types of branching strategy: either free decision variables are sequentially fixed in a predetermined order or they are chosen in a variable manner. These are called fixed-order and variable-order branching, respectively. Branching stops when the optimal solution $Z(\underline{\delta}^*)$ is found or when

$$Z_k^L \geq U \qquad \forall\, k \tag{8}$$

In these network-analysis problems, the works of Loubal (31), Murchland (32), and Halder (33) on minimum-path recomputation are often used to solve the auxiliary, algebraic problem--aside from the regular minimum-path traffic assignments (34,35). In a network in which link (m,n) is shortened from C_{mn} to C_{mn}', the auxiliary problem of tree search is to find the new shortest distance $C_*^{k\ell}$ and its corresponding route. Murchland suggests the following method of updating an existing $C^{k\ell}$ and the routing matrix in minimum-path computation:

$$C_*^{kn} = \min_m (C^{kn}, C^{km} + C_{mn}') \qquad \forall k,\ k \neq n \tag{9}$$

and

$$C_*^{k\ell} = \min_n (C^{k\ell}, C_*^{kn} + C^{n\ell}) \qquad \forall k \text{ and } \ell,\ \ell \neq k,\ \ell \neq n \tag{10}$$

Loubal's algorithm (31) can be thought of as a special case of the Murchland method generalized to more than the matrix minimum-path computation.

Neither Loubal's nor Murchland's algorithm is particularly efficient for link lengthening (or when a link is deleted from a network). Halder's method (33) of competing links specifically addresses this problem. Assume that N^m stands for the set of nodes contained in a tree built from m as the root and N^n is the set of nodes in the tree with n as the root. Now, in the general case, define L^D as the set of links the removal of which would disconnect every node of N^m from N^n. This means a minimum path from a node k in N^m to a node ℓ in N^n will contain one of the (competing) links of L^D. Updating the minimum paths after lengthening link (m,n) in L^D involves

$$C_*^{k\ell} = \min_{(rs)\epsilon L^D} (C^{kr} + C_{rs} + C^{s\ell}) \tag{11}$$

It can be seen that this link-lengthening procedure is not as efficient as the link-shortening one. One thing remains clear, however: minimum-path updates normally involves n^2 instead of n^3 arithmetic operations, where n is the number of nodes in the network. The computational savings achieved by updating is obvious.

UPPER VERSUS LOWER BOUNDING TECHNIQUES

The project vector, $\underline{\delta} = (\delta_{ij})$, whose entries δ_{ij} are 0-1s, denotes the rejection or acceptance of project ΔC_{ij}. Thus $\underline{\delta} = (0110)$ denotes the rejection of the first and last link project and acceptance of the second and third. As suggested previously, the branch-and-bound tree with a 0-1 branching rule defines the combinatorial space of $\underline{\delta}$. An interesting relation is observed between the project vector $\underline{\delta}$ and the objective function, Z:

(01) If $\underline{\delta}'$ is identical to $\underline{\delta}$ except that $\underline{\delta}'$ has more entries of 1's, then $Z' \leq Z$ or $Z' \geq Z$, corresponding to a minimizing or maximizing objective, respectively.

To see this, let us consider three states in the network-design problem as shown in Equations 1-5: (a) link (i,j) carries no flow, or (b) it carries the flow on one O-D pair k-ℓ only, or (c) it carries the flow of multiple O-D pairs. Link (i,j) belongs to one of the three states. A reduction in travel cost in (i,j) would result in link (i,j) staying in state t or going to a higher state t+k; k = 0,1,2. The change to a higher state is caused because some O-D flows find it less costly to use the reduced cost link on their paths from k to ℓ. In the case of perfectly inelastic demand, for example, the total travel cost Z can either stay the same, Z'=Z, corresponding to staying in state (a) before and after, or decrease, Z' < Z, when one or more O-D pair flows find it less costly to traverse link (i,j), corresponding to states (b) and (c). On the other hand, in the case in which demand is a function of level of service, the total amount of O-D movements will be increased because of network improvement, at a nondecreasing cost. Hence system user cost will be increased (i.e., $Z' \geq Z$) as a result.

A second observation is given between the project vector $\underline{\delta}$ and the constraint $E(\underline{\delta})$ [where $E(\underline{\delta}) = \underline{b}^T\underline{\delta} \leq B$ in the classical formulation given in Equations 1-5]:

(02) For a monotonically decreasing function $E(\underline{\delta}^T\underline{\delta})$ and a monotonically increasing function $E'(\underline{\delta}^T\underline{\delta})$, suppose $E(\underline{\delta}) \leq E_o$ or $E'(\underline{\delta}) \geq E_o$. Then a vector $\underline{\delta}'$, which is identical to $\underline{\delta}$ except that $\underline{\delta}'$ has more entries of 1's, is an infeasible solution.

To see this, take the budget $E(\underline{\delta}) = \underline{b}^T\underline{\delta}$, which is a monotonically increasing function of $\underline{\delta}^T\underline{\delta}$. At optimality, the dot product of $\underline{\delta}$ is at its maximum value consistent with the budget constraint $E(\underline{\delta}) = \underline{b}^T\underline{\delta}^* \leq B = E$. Because the cost of implementing any project is nonnegative, adding another project to a subset of projects that already uses the budget to its limit would certainly exceed the budget and become infeasible.

These two observations, together with the branching strategy, make it possible to bound and fathom. Each auxiliary problem is to update a multiple O-D minimum-path computation by an algorithm suggested by Murchland (32) or Halder (33). Because the number of candidate project links is typically only a minor fraction of the number of all links in the network and a subset of the candidate links is defined at a node (say κ of them), the number of calculations is on the order of $\kappa n^2 << n^3$. This inequality becomes quite significant when n is large, as is the case with most real-world problems. Depending on the actual tree-search algorithm, however, the size κ may vary significantly.

For the sake of clarity, the tree-search method will be illustrated in its detailed algorithmic steps, in which the following procedures are used to solve the classical minimization network-design problem outlined in Equations 1-5.

Branch-and-Bound Algorithm

Step 1

Generate the active root node, r = 1. Define for this node $\underline{\delta} = (\underline{1})$ and label it with objective function Z_1. Go to Step 4.

Step 2

If an active node j has $b_j = \underline{b}^T\underline{\delta}_j \leq B$ (i.e., node j is feasible), set upper bound $U = Z_j$. Put node j on inactive status. All active, feasible nodes i with $Z_i \geq U$ are dominated [by (01)]. Put these dominated nodes on inactive status. If there are no more active terminal nodes, terminate the algorithm. The optimal solution, or solutions, $Z_j^* = U$ has been found.

Step 3

Branch: Branch from the bounded node ℓ, creating node r + 1 to the right and r + 2 to the left. Set a free variable $\delta_{ij} = 1$ on the right branch and $\delta_{ij} = 0$ on the left branch. At node r + 1, add δ_{ij} to the set of variables with assigned values, I. Calculate $\hat{B}_{r+1} = \sum_{i \in I} b_i$. If $\hat{B}_{r+1} > B$, node r + 1 has been fathomed [by (02)] and termed inactive. Otherwise, set $Z_{r+1} = Z_\ell$. At node r + 2, solve the auxiliary problem corresponding to $\underline{\delta}_{r+2}$ to obtain Z_{r+2}.

Step 4

Bound: Out of the set of active (i.e., the lower bound Z_ℓ^L) nodes, find the node ℓ with the smallest objective function Z_ℓ. Node ℓ is the bounded node. If $r \neq 1$, set r = r + 2. Go to Step 2. An example of this algorithm is shown in the Appendix.

Branch-and-Backtrack Algorithm

Step 1

Generate the active root node 1 and set counter r = 1. Set U = ∞. Define $\underline{\delta} = (\underline{0})$. The set of free variables, F, consists of all δ_{ij}'s. The problem as defined here has exactly the flow pattern of the original network before any project implementation. Call the present objective function Z_1.

Step 2

Backtrack: Out of the set of active terminal nodes, find the node with the largest node number (i.e., the latest active terminal node). If $r \neq 1$, set r = r + 2.

Step 3

Branch: Branch from the latest active node. Create node r + 1 to the right and node r + 2 to the left. Set a free variable $\delta_{ij} = 1$ on the right branch and 0 on the left branch. At node r + 1, compute $\underline{b}^T \underline{\delta}_{r+1} = B_{r+1}$. If $B_{r+1} \geq B$, node r + 1 has been fathomed [by (02)] and termed inactive. Otherwise, declare node r + 1 active. At node r + 2, let F be the new set of free variables after δ_k has been fixed. If $\sum_{i \in F} b_i = B_{r+2} \leq B$, then node r + 2 is feasible. Solve the auxiliary problem corresponding to setting the free variable in the current $\underline{\delta}$ to unity and obtain Z_{r+2}. If $Z_{r+2} \leq U$, modify the upper bound to be $U = Z_{r+2}$. Declare node r + 2 inactive. All feasible nodes with $Z_i > U$ are dominated [by (01)]. Put these dominated nodes on inactive status. If there are no more active terminal nodes, terminate the algorithm. Optimal solution U has been found. On the other hand, if $B_{r+2} > B$, declare node r + 2 active. Go to Step 2. An example of this algorithm is also contained in the Appendix.

Parametric Branch-and-Bound

In the branch-and-bound scheme proposed previously, a parametric analysis (8,22,36) can be performed on the budget level. Sensitivity analysis can be carried out to find the range within which the budget B can vary, and the solution obtained still remains optimal.

First is discussed the procedure for finding the lowest budget B (henceforth called the budget "floor") at which the solution still remains optimal. A solution is shown to be optimal in the branch-and-bound procedure by establishing its feasibility and that it occurs at a bounded node. Suppose the optimal solution occurred at the bounded node ℓ, incurring a cost of $\underline{b}^T \underline{\delta}_\ell$, which is less than B. B can conceivably be decreased to $\underline{b}^T \underline{\delta}_\ell = B^L$ without affecting the feasibility of solution $\underline{\delta}_\ell$. Therefore B can be decreased by ΔB and the former solution would still remain optimal:

$$-(B - \underline{b}^T \underline{\delta}_\ell) \leq \Delta B \leq 0 \tag{11}$$

Second, a question can be posed: How large could B be and the solution still remain optimal? This upper limit is called the budget "ceiling," B^U. Determining the budget ceiling is more complicated than determining the budget floor because the feasibility dominance rule has been employed. Recalling the way the bounding operation was carried out, the partial solution with an objective function value Z_ℓ' closest to the optimal one $Z_\ell (Z_\ell' \leq Z_\ell)$ is obtained at the bounded node in the iteration just before the one that provides the optimum. The solution at ℓ', $\underline{\delta}_\ell'$, is clearly infeasible because, if it were feasible, it would have been accepted as the optimal solution (remember $Z_\ell' \leq Z_\ell$). To note this, $\underline{b}^T \underline{\delta}_\ell' > B$ can be written. Therefore the budget could have been expanded up to (but not set at) $B_\ell' = \underline{b}^T \underline{\delta}_\ell'$ and the current optimum would still be optimal. The amount ΔB by which B can be increased is expressed as

$$0 \leq \Delta B \leq (\underline{b}^T \underline{\delta}_\ell' - B) \tag{12}$$

If it is guaranteed that all the other nodes dominated due to feasibility reasons (call it the set G) have partial solutions $\underline{\delta}_i^P$, incurring budgets greater than B_ℓ', that is,

$$\underline{b}^T \underline{\delta}_i^P > \underline{b}^T \underline{\delta}_\ell \quad i \in G \tag{13}$$

then Z_ℓ would be the first optimal solution encountered as B is incrementally expanded. For this reason, to maintain optimality for the current solution, B definitely cannot be increased by more than ΔB as prescribed in Equation 12. On the other hand, if Equation 13 is not guaranteed, a second-best solution could conceivably be found in the set G. Under these conditions, only a weak upper bound could be obtained by taking $\underline{b}^T \underline{\delta}_\ell$. This section can be summarized by saying that

$$-(B - \underline{b}^T \underline{\delta}_\ell) \leq \Delta B \leq \min \left[(\underline{b}^T \underline{\delta}_\ell' - B), \min_{i \in G} (\underline{b}^T \underline{\delta}_i^P - B) \right] \tag{14}$$

Discussion of Solution Methods

Two solution methods, branch-and-bound and branch-and-backtrack, were outlined in the previous sections to solve the network investment problem. The branch-and-bound scheme adopts a strategy of branching from the lowest bound. The root node accepts all projects [$\underline{\delta} = (\underline{1})$] and "rejects" projects one by one during branching. Computationally, each auxiliary problem may involve using the link-shortening algorithm quite a few times (up to κ times, where κ is

the number of 1's in $\underline{\delta}$). For this reason, Halder's method of lengthening links may be more applicable here because only one additional link needs to be lengthened at a time.

The computer storage required to retain the intermediate information may have to be quite large. Programming the branch-and-bound algorithm may not be easy either. It requires a sophisticated data structure to jump efficiently from one node to another and to regenerate solution information at nodes not recently visited. But the greatest difficulty lies in controlling the number of terminal nodes. Storage space, rather than solution time, is the key constraint on this method. Because of the greater complexity of programming, data manipulation, and branching node choice, the execution speed is slower than for the branch-and-backtrack method.

The branch-and-backtrack method keeps on branching from the latest active node. In the present algorithm, a root node of all zeros [$\underline{\delta} = (\underline{0})$] is used to start and projects are accepted one by one during branching. Computationally, this branching rule saves solving quite a few auxiliary problems (five compared with nine in the problem worked out in the Appendix) because the auxiliary problem at a left-hand node need not be evaluated until feasibility is encountered. Also, each auxiliary problem defined in a branch-and-backtrack tree typically has fewer entries of 1's. The number of calculations in a link-shortening algorithm, $\kappa\eta^2$, is smaller because κ is not as big as most of the κ's found in the branch-and-bound auxiliary problems.

Inside the computer, a pushdown list can be used. The entry on the top would correspond to the most recent active element. Each time branching is carried out, the new problems are placed on top of the stack. Each time branching is to be performed, elements from the top of the stack are examined. If active, new elements will be added to the stack, corresponding to the new problems generated by branching. If not active, the element will be discarded until an active element is encountered. The length of the stack will be proportional to the length of the longest path directed away from the root of the tree.

The branch-and-bound procedure of branching from the lowest bound obviously gives the best criteria for choosing the next node to branch from, in that the node chosen is more likely to have an optimal solution at its successor than the node automatically chosen by branch-and-backtrack. The scheme of branch-and-bound thus allows sensitivity analysis to be performed on the budget as outlined earlier, and this author finds it infeasible to perform an equivalent sensitivity analysis on the solution obtained by the branch-and-backtrack method. On the other hand, branch-and-backtrack tends to arrive at a feasible solution fast, even though it may be far from optimal.

There is a certain similarity between the tree-search scheme proposed here and the unimodal function search discussed by Mitten (25). In Figure 1 is depicted the trade-off relationship between optimality and feasibility. On the left end of the z-axis are solutions with low Zs, yet most of them are infeasible solutions. On the right portion of the axis are high Zs, but they tend to be feasible solutions. In the center portion will lie the optimum solution that satisfies optimality and feasibility. The branch-and-bound procedure generates solutions from the lowest bound. It approaches the solution from the left portion of the z-axis. The branch-and-backtrack method emphasizes getting feasible solutions fast. It operates from the right portion of the z-axis, edging onto the center portion. If the computer time available does not permit the execution of the tree-search scheme to completion, chances are that branch-and-backtrack will give at least a feasible solution and an upper bound. Branch-and-bound may just give an infeasible solution and a lower bound.

A DOUBLE-BOUNDING TECHNIQUE

The branch-and-bound and branch-and-backtrack algorithms are flexible enough to address a generalized cost function; a demand function; and a user- and system-optimizing, minimization, or maximization objective. There is one unsatisfying element about the solution method, however, and that is the difficulty of finding strong bounds, particularly both an upper and a lower bound. A strong upper and lower bound are critical to improving computational efficiency. Preliminary research has led to an algorithm that is discussed hereafter. Again, a minimization objective of the network-design formulation is assumed for convenience in the algorithmic steps. The design of the algorithm is motivated by works of Chan (37), Billheimer and Gray (38), Magnanti and Wong (39), and Ruiter (40).

Preliminary Step

Set up a state-stage diagram, as shown in Figure 2, in which the rows correspond to the O-D pairs and the columns correspond to the number of algorithmic iterations. The O-D demands corresponding to level of service m are also sketched in as $\pi^m_{k\ell}$, corresponding to the amount of induced demand increments. Initialize m = 0. Solve the first auxiliary problem by performing a traffic assignment for the network (if one is not already available), yielding the upper bound objective function $U^m = Z_U$. A parallel assignment is

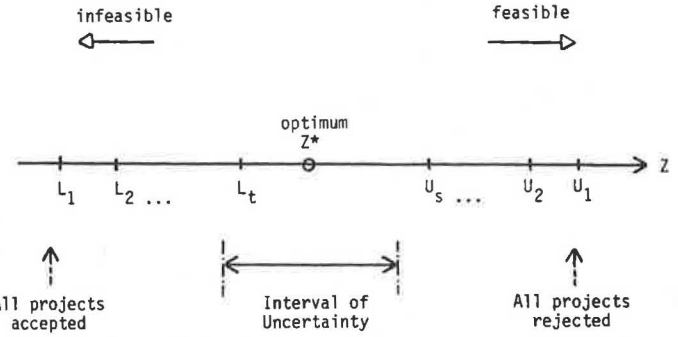

FIGURE 1 Bounding from above and below.

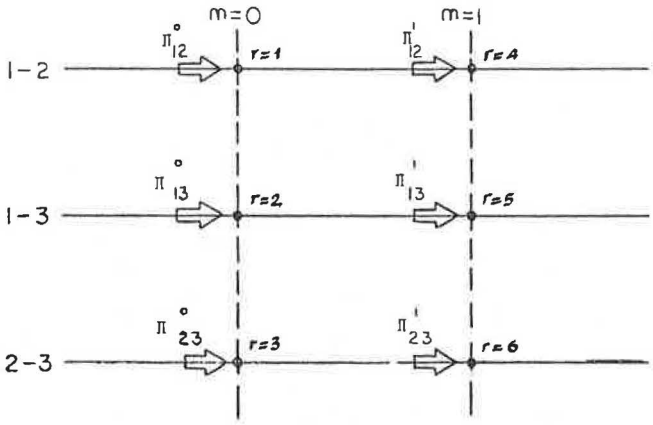

FIGURE 2 Demand function as represented in the state-stage diagram.

performed on a network where all the projects in the specified set R have been implemented, yielding the lower bound objective function $L^m = Z_L$. Set S = R and T = R.

Step 1: Link Insertion

Set m←m + 1. Take each node m-r in the state-stage diagram as the active node, starting from the top in a fixed-order sequence. From the set of project candidates, S, a subset of projects $I = [\delta_i]$ is selected. The improvement in the objective function $\Delta Z_j^U(m)$ over the upper bound is obtained by updating the traffic assignment for each of the $\delta_j \varepsilon I$. The project that results in the best improvement δ_j is accepted and set to unity. The corresponding objective function $U_m = U_{m-1} - \Delta Z_j^U(m)$ and budget level $B_U(m) = \sum_{j\varepsilon S} \delta_j b_j$ are then computed. Project δ_j is then removed from the set S for further consideration.

Step 2: Link Removal

For the same nodes m-r, a project δ_k (if any) is selected from the subset I if the elimination of it from the network (i.e., setting $\delta_k = D$) results in the minimal (but nonzero) degradation of the objective function, and the degradation $\Delta Z_k^L(m)$ has to be less than the improvement $\Delta Z_j^U(m)$ [i.e., $\Delta Z_k^L(m) < \Delta Z_j^U(m)$]. The corresponding objective function, $L_m = L_{m-1} + \Delta Z_k^L(m)$, and the budget level, $B_L(m) = \sum_{k\varepsilon T} \delta_k b_k$, are computed. Project δ_k is then removed from T.

Step 3: Termination Criteria

When both the upper and lower bound solutions are feasible [i.e., $B_L(m)<B$ and $B_U(m)<B$] or both sets S and T are empty, stop. A local optimal solution $Z^* = \min(U_m, L_m)$ has been found, with the corresponding projects S or T and budget level. Otherwise, after all the nodes m-r have been scanned and become inactive, go back to Step 1. An example of this double-bounding algorithm, to accompany the branch-and-bound and branch-and-backtrack examples, is shown in the Appendix.

CONCLUSION

This paper serves as a brief review of tree-search methods as applied to transportation network design. The example problem is formulated as a user-optimizing, nonlinear, multicommodity, fixed-charge-type integer program. The integer program is solved by two approaches, branch-and-bound and branch-and-backtrack. Postoptimality procedures are used to solve the auxiliary problem generated by the tree-search schemes. The concept of parametric branch-and-bound is sketched, showing that sensitivity analyses can be performed as part of the algorithm. Finally, a comparison is made between the two solution methods. This results in the design of a double-bounding algorithm.

It is observed that from the computation and computer programming point of veiw, the branch-and-backtrack algorithm is more efficient than the branch-and-bound algorithm. Branch-and-backtrack provides feasible solutions quite early in the computation. It approaches the optimal solution via an upper-bound pruning rule. Branch-and-bound gives feasible solutions only at the final phase of the algorithm, approaching the optimal solution mostly from the lower bound. Parametric sensitivity analysis can be performed with the branch-and-bound algorithm, whereas the author sees no way to do the same with branch-and-backtrack.

The proposed double-bounding algorithm has the promise of being computationally more efficient. The solution so obtained is, nevertheless, merely a local optimum. More research is needed in its refinement. Available information substantiates the value of tree-search methods in solving a number of transportation network-design problems with typically ill-behaved nonanalytical properties.

It should be noted that the tree-search algorithms presented here are based on the monotonicity properties (01) and (02), which essentially assume that travel congestion in vehicle-minutes is reduced for perfectly inelastic demands as links are added to the network. Likewise, it is assumed that the number of O-D movements is increased on an improved network for downward-sloping demand functions. Recent findings about the Braess' paradox by Steinberg and Zangwill (41) show that, should all routes used before the addition of the new link continue to be used, travel congestion for the inelastic demand case may be worsened as a result of link addition. This would in some cases violate the first of the two monotonicity properties on which tree-search solution algorithms are built and raises serious doubts over the wealth of literature on tree-search.

However, another recent finding is of interest. Pearman (42) found out that network-design problems are "rich in suboptimal solutions." Because there are other concerns in transportation planning aside from an "optimal" solution to the network-design problem, any improved network design, even though only locally optimal, may be useful in practice. Following this line of argument, a strong case can still be made for using the tree-search algorithms presented here as a computational tool to get better network designs; although the researcher would necessarily have to be humble in claiming that the algorithms are panaceas for solving all transportation woes.

REFERENCES

1. J.L. Kennington. A Survey of Linear Cost Multicommodity Network Flows. Operations Research, Vol. 26, No. 2, 1978, pp. 209-236.
2. L.D. Charnes and W.W. Cooper. Multicopy Traffic

Network Models. Symposium on The Theory of Traffic Flow, General Motors Research Laboratory, Warren, Mich., 1959, pp. 85-96.
3. S.C. Defermos. An Extended Traffic Assignment Model with Applications to Two-Way Traffic. Transportation Science, Vol. 5, No. 1, 1971, pp. 366-389.
4. G. Dantzig, R. Harvey, Z. Lansdowne, D. Robinson, and S. Maier. Formulating and Solving the Newtork Design Problem by Decomposition. Transportation Research, Vol. 13B, 1978, pp. 5-17.
5. L.J. LeBlanc and M. Abdulaal. An Efficient Trial Approach to the Urban Road Network Design Problem. Computers and Mathematics with Application, Vol. 5, 1979, pp. 11-19.
6. H.M. Weingartner. Mathematical Programming and the Analysis of Capital Budgeting Problems. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1963.
7. Y. Chan. Optimal Travel Time Reduction in a Transport Network: An Application of Network Aggregation and Branch and Bound Techniques. Report R69-39. Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, 1969.
8. A.S. El-Aroud. A Road Development Model for Developing Countries. Ph.D. dissertation. Pennsylvania State University, University Park, 1980.
9. M.L. Funk and E.A. Tillman. Optimal Construction Staging by Dynamic Programming. Journal of the Highway Division, ASCE, 1968, pp. 255-265.
10. G. Bergendahl. A Combined Linear and Dynamic Programming Model for Interdependent Road Investment Planning. Transportation Research, Vol. 3, 1969, pp. 211-228.
11. L.D. Chapman. Investing in Regional Highway Networks. Journal of the Transportation Engineering Division, ASCE, 1973.
12. T.T. Shen. Application of Traffic Assignment and Network Aggregation in an Urban Area. Master's Thesis. National Taiwan University, Taipei, Taiwan, Republic of China, 1981.
13. D. Johnson, J. Lenstra, and A.R. Kan. The Complexity of the Network Design Problem. Networks, Vol. 8, 1978, pp. 279-285.
14. R. Dionne and M. Florian. Exact and Approximate Algorithms for Optimal Network Design. Networks, Vol. 9, 1979, pp. 37-59.
15. R.T. Wong. Worst-Case Analysis of Network Design Problem Heuristics. Journal of Algebraic and Discrete Methods, Society for Industrial and Applied Mathematics, Vol. 1, 1980, pp. 51-63.
16. D. Dubois, G. Bel, and M. Llibre. A Set of Methods in Transportation Network Synthesis and Analysis. Journal of the Operational Research Society, Vol. 30, No. 9, 1979, pp. 798-808.
17. A.H. Land and A. Doig. An Automatic Method of Solving Discrete Programming Problems. Econometrica, Vol. 28, 1960, pp. 497-520.
18. J.D.C. Little, K.C. Murty, D.W. Sweeney, and C. Karel. An Algorithm for the Travelling Salesman Problem. Operations Research, Vol. 11, 1963, pp. 972-989.
19. E. Balas. An Additive Algorithm for Solving Linear Programs with O-1 Variables. Operations Research, Vol. 13, 1965, pp. 517-549.
20. E. Balas. A Note on the Branch-and-Bound Principle. Operations Research, Vol. 16, No. 2, 1968, pp. 422-445. Errata in Vol. 16, No. 4, p. 886.
21. E.L. Lawler and D.E Wood. Branch and Bound Methods: A Survey. Operations Research, Vol. 14, No. 4, 1966, pp. 699-719.
22. J.D. Ichbiah. Connectivity Analysis and Branch and Bound Methods. Ph.D. dissertation. Massachusetts Institute of Technology, Cambridge, 1967.
23. F. Ochoa and A. Silva. Optimum Project Addition in Urban Transportation Networks via Descriptive Traffic Assignment Models. Research Report R68-44. Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, 1968.
24. A.J. Scott. The Optimal Network Problem: Some Computational Procedures. Transportation Research, Vol. 3, 1969, pp. 201-210.
25. L.G. Mitten. Branch-and-Bound Methods: General Formulation and Properties. Operations Research, Vol. 18, No. 1, 1970, pp. 24-34.
26. H.H. Hoc. A Computational Approach to the Selection of an Optimal Network. Management Science, Vol. 19, No. 3, 1973, pp. 488-498.
27. D.E. Boyce, A. Farhi, and R. Weischedel. Optimal Network Problems: A Brand-and-Bound Algorithm. Environment and Planning, Vol. 5, 1973, pp. 519-533.
28. P.A. Steenbrink. Optimization of Transport Networks. John Wiley & Sons, Inc., New York, 1974.
29. K. Chen and K.P. Jarboe. Large Scale Decision Tree Optimization by Branch and Bound Methods. Large Scale Systems, Vol. 1, No. 2, 1980, pp. 117-128.
30. H. Hoorzahedy and M.A. Turnquist. Approximate Algorithms for the Discrete Network Design Problem. Transportation Research, Vol. 16B, No. 1, 1982, pp. 45-55.
31. P.S. Loubal. A Network Evaluation Procedure. In Highway Research Record 205, HRB, National Research Council, Washington, D.C., 1967, pp. 96-109.
32. J.D. Murchland. A Fixed Matrix Method for all Shortest Distances in a Directed Graph and for the Inverse Problem. Transport Network Unit LBS-TNT-91. London Business School, London, England, 1969.
33. A.K. Halder. The Method of Competing Links. Transportation Science, Vol. 4, No. 1, 1970.
34. D. Van Vliet. Improved Shortest Path Algorithms for Transportation Networks. Transportation Research, Vol. 12, 1978, pp. 7-20.
35. N. Deo, Y.B. Yoo, and M.J. Quinn. Parallel Processing for Some Large-Scale Network Optimization Problems. University Research Program, U.S. Department of Transportation, 1983.
36. F. Ochoa-Rosso. Applications of Discrete Optimization Techniques to Capital Investment and Network Synthesis Problems. Research Report R68-42. Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, 1968.
37. Y. Chan. Configuring a Transportation Route Network via the Method of Successive Approximation. Computers & Operations Research, Vol. 1, 1974, pp. 385-420.
38. J.W. Billheimer and P. Gray. Network Design with Fixed and Variable Cost Elements. Transportation Science, Vol. 7, No. 1, 1973.
39. T.L. Magnanti and R.T. Wong. Network Design and Transportation Planning: Models and Algorithms. Transportation Science, Vol. 18, No. 1, 1984.
40. E.R. Ruiter. A Prototype Analysis: Search and Choice in Transportation Systems Planning. Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, Vol. 2, 1968.
41. R. Steinberg and W.I. Zangwill. The Prevalence of Braess' Paradox. Transportation Science, Vol. 17, No. 3, 1983, pp. 301-318.
42. A.D. Pearman. The Structure of the Solution Set to Network Optimization Problems. Transportation Research, Vol. 13B, 1979, pp. 81-90.

APPENDIX--SOLUTION OF AN EXAMPLE PROBLEM

Consider the following example problem with perfectly inelastic demands (Figure A-1):

$$\min Z = \sum_{k=1}^{4}\sum_{\ell=1}^{4} (4 - \delta_1)\, x_{12}^{k\ell} + (2 - \delta_2)\, x_{41+(2-\delta 3)}^{k\ell}\, x_{24}^{k\ell} + (4 - \delta_4)\, x_{34}^{k\ell} + 4x_{23}^{k\ell} + 7x_{42}^{k\ell} + 7x_{14}^{k\ell} + x_{43}^{k\ell}$$

such that

$$\sum_i x_{ip}^{k\ell} - \sum_j x_{pj}^{k\ell} = \begin{cases} -1 \text{ if } p = k \\ 1 \text{ if } p = \ell \\ 0 \text{ otherwise} \end{cases} \qquad k \neq \ell,\ \forall p$$

$$\delta_1 + 2\delta_2 + 2.5\delta_3 + 1.5\delta_4 \leq 4$$

$x_{ij}^{k\ell} > 0$ only if (i,j) is on the shortest path from k to ℓ

$x_{ij}^{k\ell}$ = positive integers; $\delta_r = (0,1)$ where a shorthand notation δ_1 has been used to denote δ_{12}, δ_2 for δ_{41}, δ_3 for δ_{24} and δ_4 for δ_{34}.

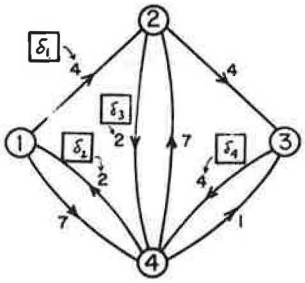

FIGURE A-1 Example network.

Branch-and-Bound

For this network design problem, the branch-and-bound algorithm is stepped through in detail as a directed tree. The reader should refer to Figure A-2 as he goes through the algorithm, where the node numbers correspond to the sequence in which the algorithm is carried out.

Branch-and-Backtrack

The same network design example will be used to illustrate the branch-and-backtrack algorithm. The reader should refer to the directed tree shown in Figure A-3 when he goes through the algorithm steps.

Double-Bounding Algorithm

Again, using the same example, these algorithm steps are performed

m = 0

In this preliminary step, a state-stage diagram is generated with 12 rows and an indefinite number of columns. Traffic assignments yield U° = 55 vehicle-minutes when all projects are rejected and L° = 37 with all projects implemented. $S = \{\delta_1, \delta_2, \delta_3, \delta_4\}$, with all δ's equal to 0. $T = \{\delta_1, \delta_2, \delta_3, \delta_4\}$, with all entries set at 1.

m = 1

Link Insertion

From the entire set of link-improvement candidates, links are improved by setting each δ_k to 1 in the upper-bound network. The corresponding system travel cost (Z) in vehicle-minutes is calculated using a link-shortening procedure such as Murchland's.

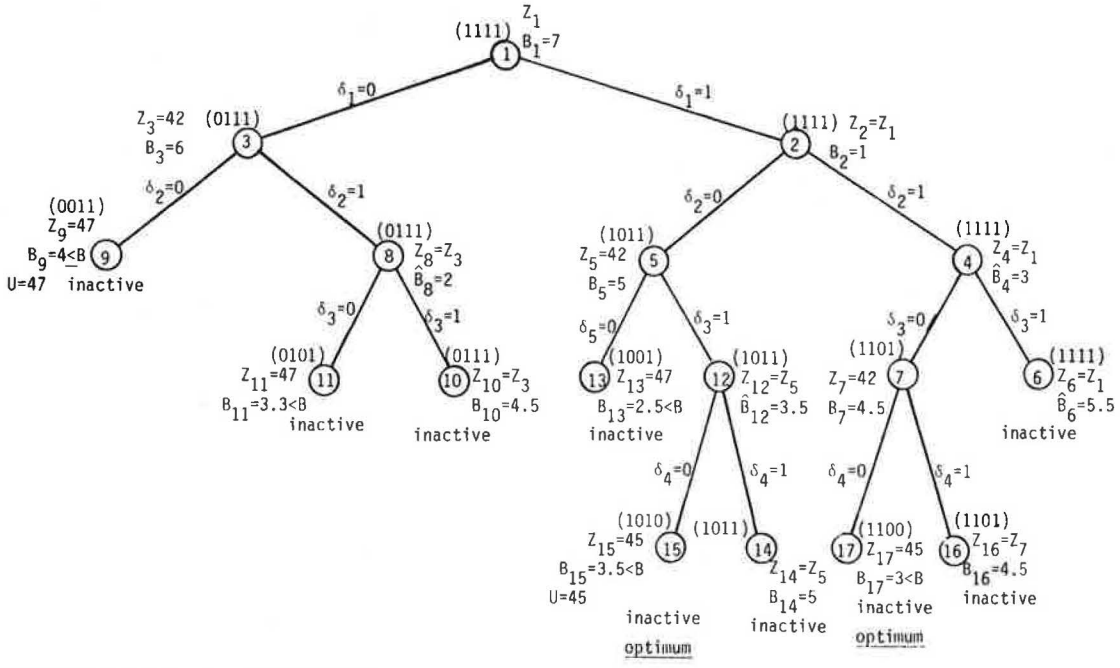

FIGURE A-2 Branch-and-bound example.

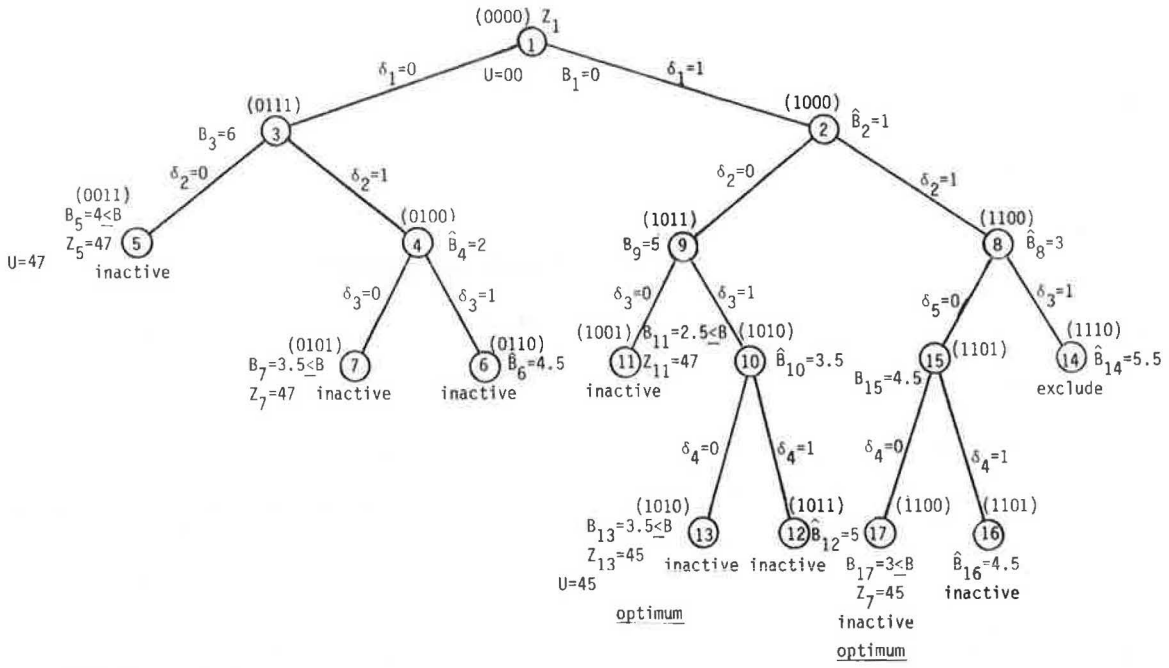

FIGURE A-3 Branch-and-backtrack example.

δ_2 results in the best improvement in Z of 7 vehicle-minutes. Hence $U_1 = 55 - 7 = 48$, $B_U(1) = 2 < B = 4$. δ_2 is then removed from the candidate set S.

Link Removal

Similarly, link-improvement candidates are removed from the network by setting each δ_k to 0 in the lower-bound network. The corresponding system travel cost (Z) is computed using a link-lengthening algorithm such as Halder's. The removal of candidate δ_2 results in the minimal degradation of Z, with the amount of degradation $\Delta Z_2^L(1) = 5 < \Delta Z_2^U(1) = 7$. Now $L_1 = 37 + 5 = 42$ and $B_L(1) = 5$ and project δ_2 is removed from T.

m = 2

Link Insertion

This second iteration inserts δ_1 into the upper-bound network, resulting in $U_2 = 44$, $B_U(2) = 3$, which is less than B = 4. Also, S now consists of δ_3 and δ_4 only, both at values of zero, with δ_1 and δ_2 set at unity.

Link Removal

Similarly, δ_4 is removed from S_2, resulting in $L_2 = 45$, $B_L(2) = 3.5 < B = 4$. Now T consists of δ_1 and δ_3 only, both at unity, and δ_2 and δ_4 are set at zero.

Termination

Because both upper- and lower-bound solutions are feasible, a local optimum $Z^* = \min(44,45) = 44$ vehicle-minutes is obtained with the corresponding projects, δ_1 and δ_2, implemented. Comparing the results with those of tree search, it is found that the solution is, indeed, a global minimum.

Publication of this paper sponsored by Committee on Application of Economic Analysis to Transportation Problems.