Transportation-Network Design Problem: Application of a Hierarchical Search Algorithm

YUPO CHAN, T. STEVEN SHEN, AND NIZAR M. MAHABA

Two variants of a network design problem are solved by application of the tree search method. The first formulation aims to reduce a specified vehicle-minutes of traffic congestion at the least possible budget expenditure, and the second minimizes traffic congestion for a given budget. Both involve system-optimizing traffic assignment models with multipath flows. The solution method consists of network abstraction, tree search, and network disaggregation—collectively referred to as the "hierarchical search algorithm." It is shown that such an algorithm reduces the search space by reducing the number of nodes and links and providing a tighter bound during the tree search. It also groups detailed links according to the function they perform—whether it be access/egress, line-haul, bypass, or internal circulation. However, the algorithm yields only a suboptimal solution, the quality of which is measured by an error function. The metropolitan network of Taipei, Taiwan, Republic of China, is used as a case study to verify some of the algorithmic properties, confirming its role in real-world applications. Finally, the performance of the algorithm, which is based on network abstraction, is favorably compared with a network-extraction network-design model.

Theoretical advances in the last two decades have significantly improved our understanding of network traffic flow. Numerous equilibration models have been put forth under both system-optimizing and user-optimizing assumptions. In spite of advances in computational hardware and software, however, the network-design problem is an NP-hard problem that defies efficient solution techniques (*1*). This is a particularly acute problem in practice, where the size of networks can easily go into hundreds of nodes and links (as in our case study later). No practical solution algorithms exist today to tackle such problems satisfactorily. Difficulties still arise, for example, in solving the network design problem exactly, because it is computationally demanding to solve a user-optimized flow pattern at each step of a system-optimizing search process in the presence of Braess's paradox.

There have been several attempts to address this NP-hard problem by reducing the size of the network, which tends to cut the computational requirement exponentially. For example, network extraction techniques have been practiced for a long, long time to cut down the size of a network design problem. The approach calls for removing "insignificant" nodes and links from a network, leaving only the "important" topological features (*2,3*). However, in spite of carefully designed controlled experiments conducted during the past 20 years (*2,4*), such procedures are still heuristic in nature, often resulting in unpredictable accuracies.

In lieu of extraction, network abstraction has been examined as an alternative to cut down on dimensionality (*5,6*). In this approach, nodes and links are aggregated together to reduce network size. Partial success has been reported in placing error bounds on a limited class of transportation problems, typically variants of the classical Hitchcock transportation/assignment model (*7,8*).

Continuous equilibrium network design formulations have also been proposed. Instead of discrete node-arc representation, improvement variables are continuous (*9*). Some computational gains have been reported, even for user-optimizing traffic assignments.

The preceding aggregation efforts, although improving our ability to solve larger problems, are not quite enough—as pointed out already (*10*). Recent attempts have been made to obtain approximate solutions to both discrete and continuous network design problems. Wong (*1*), for example, revisited Scott's seminal work on discrete network design heuristics and placed worst-case analyses on the computational procedure. Suwansirikul et al. (*11*), on the other hand, suggested a heuristic for finding an approximate solution to the continuous user-optimizing network design model.

To summarize, much work remains to be done in the classic problem of network design. An obvious void is in the abstraction of a realistic transportation network (instead of the Hitchcock assignment problem) and in the placement of error bounds on the corresponding network design problem as we disaggregate back to the original problem.

Here a network design problem is formulated as a hierarchical mathematical program. The original network is abstracted into an aggregate network, thus reducing the number of nodes and links in the process (*5,12*). A tree search is then performed in the aggregate network, which serves as a proxy for the detailed network (*3,13*). The resulting network investment strategy is then disaggregated back to the original, detailed network for implementation (*14*), with a statement on the quality of the approximate solution.

PROBLEM FORMULATION

We state here the first of two network design problems, in which a lowest budget expenditure objective function is formulated as follows:

$$\text{Minimize } B = \sum_i b_i y_i = \bar{b}^T \bar{y} \quad (1)$$

Y. Chan, Department of Operational Sciences, School of Engineering (ENS), Air Force Institute of Technology, Wright-Patterson, Ohio 45433-6583. T. S. Shen, Barton-Aschman Associates, 1133 15th Street, N.W., Washington, D.C. 20005. N. M. Mahaba, 25 Hamadan Street, Giza, Egypt.

where b_t is the budget expenditure for the tth project (performed on link i,j) and

$$y_t = \begin{cases} 1 \text{ if project } t \text{ is implemented} \\ 0 \text{ otherwise} \end{cases} \quad (2)$$

Congestion cannot be allowed to exceed a certain level E_o:

$$E(\bar{y}) \leq E_o \quad (3)$$

where $E(\bar{y})$, in vehicle-minutes, is the result of a system-optimizing traffic assignment. Thus the model is more suited for system-control applications than for evaluation purposes, and y_t's are best interpreted as traffic control strategies to effect an overall improvement of areawide traffic congestion.

By way of definition

$$E(\bar{y}) = \min \sum_{kl} \sum_{ij \in R^{kl}} [f_{ij}(x_{ij}) - \Delta f_{ij}(x_{ij}) y_{ij}] x_{ij}^{kl} \quad (4)$$

is the total congestion after link improvements with the node-arc incidence matrix:

$$\sum_i x_{ip}^{kl} - \sum_j x_{pj}^{kl} = \begin{cases} -v^{kl} & \text{if } p = k \\ v^{kl} & \text{if } p = l \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

and the link-flow "bundling" equation

$$x_{ij} = \sum_{kl} x_{ij}^{kl} \quad (6)$$

where

R^{kl} = the set of links en route from origin k to destination l;
f_{ij} = a convex travel cost function of nonnegative link flow x_{ij}, constrained by a limiting capacity c_{ij};
Δf_{ij} = the improvement in link ij consisting of either travel cost reduction or capacity expansion or both;
x_{ij}^{kl} = nonnegative integer, standing for the flow from origin k to destination l in link (i,j);
y_{ij} = the same as y_t where t is specified for link ij; and
v^{kl} = the origin-destination demand.

Although an alternate formulation will be offered in sequel, recapitulated below are the basic assumptions in our network design models throughout this paper:

1. Travel demand is fixed for each origin-destination (O-D) pair and
2. System-optimizing equilibration procedure is employed.

Through network abstraction (*5,15–17*), we wish to simplify the preceding optimization by reducing the number of links and nodes. One collapses v^{kl} into V^{KL} through zonal aggregation where

$$\sum_{\substack{k \in K \\ l \in L}} v^{kl} = V^{KL} \quad (7)$$

In other words, adjacent zones k are grouped into aggregate zone K and likewise l into L. Finally, the links (i,j)s are aggregated into composite links (I,J)'s with the corresponding travel times, $\hat{F}_{IJ}$, at flow volumes x_{ij} and X_{IJ}, respectively:

$$f_{ij}(x_{ij}) \rightarrow \hat{F}_{IJ}(X_{IJ}) \qquad \forall (i,j) \quad (8)$$

We can now rewrite the preceding network design formulation (Expressions 1 through 8) by replacing every symbol with a capitalized, underlined letter, or Greek letter, converting it from the detailed space to the aggregate space:

Objective function: $B \rightarrow \underline{B}$ with $b_m \rightarrow \beta_m$ (9)

System travel cost function: $E(\bar{y}) \rightarrow \underline{E}(\bar{Y})$ (10)

Traffic flow: $x \rightarrow X$ (11)

The tree search is now carried out in the abstracted network instead of the detailed one, resulting in an optimal solution consisting of link improvements $\{\Delta \hat{F}_{IJ}\}$, rather than $\{\Delta f_{ij}\}$. Finally, a disaggregation method has to be employed to convert each of these link improvements back to the detailed network:

$$\Delta \hat{F}_{IJ} \rightarrow \{\Delta f_{ij}\} \qquad \forall (I,J) \quad (12)$$

The state of the art in network aggregation, particularly in the context of network design, is still quite rudimentary, as alluded to earlier. In the words of Zipkin (*7*):

> [E]ven with computational experience and good software, we do not envision universally appropriate procedures for aggregation. Rather, modellers will have to combine . . . techniques and judgement to suit the problem at hand.

Below, we show a network abstraction procedure that satisfies our specifications outlined by Equations 7 through 12 for a typical transportation network design problem.

NETWORK AGGREGATION ALGORITHM

As mentioned, a network abstraction procedure typically starts with zonal aggregation. In transportation analysis, the grouping of contiguous nodes k and l together is more often than not decided exogenously, mainly by political, geographical, and other considerations. This is distinctly different from scientifically motivated "error-bound" procedures that require that "topologically similar" nodes be aggregated together (*7*)—a process that may require "regrouping" a posteriori for the express purpose of tightening error bounds.

Although zonal aggregation can be accomplished quite readily here via Equation 7, link aggregation needs some explanation. According to Chan (*5,15,16*), link aggregation can be performed in three phases after an initial traffic assignment is made in the detailed network.

Phase 1. Categorization

The links aligned along the minimum paths between each detailed O-D pair, R_{kl}, are categorized first according to groupings, as illustrated in Figure 1. We identify two general classes of flow paths: (a) interzonal flow paths, such as that from node 2 to 7, which goes through aggregate zones I, II,

and III, and (b) intrazonal flow paths, such as that from node 2 to 3 where the entire path is contained in aggregate zone I.

The categorization of links in a path is then broken down into aggregate link classifications according to where the flow path is coming from and where it is leading. For example, in Figure 1, the flow path 2–7 is broken down in the following manner: The first category contains detailed links that carry the egress flow from aggregate zone I to II, which we denote as $E^{I\,II}$. Referring to Figure 1 again, it is found that links (2,1) and (1,3) are the detailed egress links that fall into this category. As a result, the set $E^{I\,II}$ contains (2,1) and (1,3) as elements. The second category contains detailed links that carry the line-haul traffic from aggregate zone I to zone II: $L^{I\,II}$. It corresponds to detailed link (3,5). Similarly, the bypass link from I to III—$^{I}B^{III}$—contains (5,4). The line-haul link $L^{II\,III}$ contains (4,7), and, finally, the *access* link $A^{II\,III}$ is made up of node 7 only. The reader may notice that each column in the summary Table 1 corresponds to an aggregate link F_a^{IJ}, each identified by IJ and aggregate link type a such as line-haul, access, and so on. In the case of a bypass link, an additional superscript specifies the zone in which the line-haul traffic passes through, $^{K}F_a^{IJ}$.

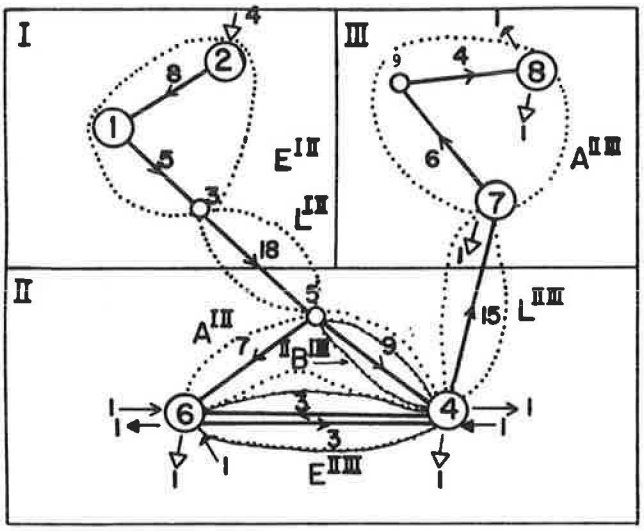

FIGURE 1 Example network.

In the entries of Table 1, the links on each path from origin k to destination l are partitioned into groups $|a_{kl}|$ according to whether they serve a line-haul, access/egress, or intraflow function:

$$|a_{kl}| \leftarrow \{(i,j)\}_{kl} \tag{13}$$

As suggested earlier, an intraflow is the traffic that originates and terminates within an aggregate zone. An example can be found in aggregate zone II between nodes 4 and 6. Unlike interzonal flows, aggregate links that carry intraflows in zone I are simply identified by F_I.

TABLE 1 LINK AGGREGATION

Y_1: $\Delta\hat{L}^{I\,II}=2$ Y_2: $\hat{A}^{I\,II}=4$ Y_3: $\Delta^{II}\hat{B}^{I\,III}=1$

$b_1=2$ $b_2=2.5$ $b_3=1$

O-D Demands	$E^{I\,II}$	$L^{I\,II}$	$A^{I\,II}$	$E^{II\,III}$	$^{II}B^{I\,III}$	$L^{II\,III}$	$A^{II\,III}$
$v^{2\,4}=1$	$f_{2\,1}+f_{1\,3}=13$	$f_{3\,5}=18$	$f_{5\,4}=9$				
$v^{2\,6}=1$	$f_{2\,1}+f_{1\,3}=13$		$f_{5\,6}=7$				
$v^{2\,7}=1$	$f_{2\,1}+f_{1\,3}=13$	$f_{3\,5}=18$			$f_{5\,4}=9$	$f_{4\,7}=15$	$f_7=0$
$v^{2\,8}=1$	$f_{2\,1}+f_{1\,3}=13$	$f_{3\,5}=18$			$f_{5\,4}=9$	$f_{4\,7}=15$	$f_{7\,9}+f_{9\,8}=10$
$v^{6\,8}=1$				$f_{6\,4}=3$		$f_{4\,7}=15$	$f_{7\,9}+f_{9\,8}=10$
Link times $^{K}\hat{F}^{IJ}$	$\hat{E}^{I\,II}=13$	$\hat{L}^{I\,II}=18$	$\hat{A}^{I\,II}=8$	$\hat{E}^{II\,III}=3$	$^{II}\hat{B}^{I\,II}=9$	$\hat{L}^{II\,III}=15$	$\hat{A}^{II\,III}=6\ 2/3$

	I_{II}	i_{II}
$v^{4\,6}=1$	$f_4=0$	$f_{4\,6}=3$
$v^{6\,4}=1$	$f_{6\,4}=3$	$f_4=0$
Link times	$\hat{I}_{II}=1\ 1/2$	$\hat{i}_{II}=1\ 1/2$

Phase 2. Summation

In this phase, the travel times of a serial string of links belonging to the same aggregate link grouping $|a_{kl}|$ are summed. Referencing Figure 1 again as an example, in the $E^{\text{I II}}$ grouping and for flows between 2 and 7, we find links (2,1) and (1,3) with travel times $f_{2\ 1} = 8$ and $f_{1\ 3} = 5$, respectively. The summation phase calls for the addition of these two elements, producing the aggregate link time $\hat{F}^a_{kl} = 13$, which is tabulated in Table 1. In general, detailed link times f_{ij} in the chain from detailed zone k to zone l are summed to become $\hat{F}^a_{kl}$:

$$\hat{F}^a_{kl} = \sum_{ij \in |a_{kl}|} f_{ij} \tag{14}$$

Phase 3. Weighting

Up to the last step of our aggregation procedure, we have in general several aggregate-time quantities $\hat{F}^a_{kl}$ under each aggregate link F^{IJ}_a (or ${}^K F^{IJ}_a$), as illustrated in Table 1. Our final objective, as the reader may expect, is to derive a single link time for each aggregate link F^{IJ}_a. This is accomplished by computing the weighted average (or convex combination) of chains in parallel, where the detailed O-D flows are used as weights (normalized by the total flow volume on the aggregate link). The weighted average becomes the required link travel time $\hat{F}_a$.

Take the example of the F^a_{kl}'s under $A^{\text{II III}}$ in Table 1. There is one trip on $F^a_{2\ 7}$, one trip on $F^a_{2\ 8}$, and one trip on $F^a_{6\ 8}$, resulting in a total of three trips on the aggregate link $A^{\text{II III}}$. Weighting $\hat{F}^a_{2\ 7}$, $\hat{F}^a_{2\ 8}$ and $\hat{F}^a_{6\ 8}$ equally by ⅓, ⅓, and ⅓, respectively, we obtain 6⅔, which is the link time for $A^{\text{II III}}$. In general,

$$\hat{F}_a = \sum_{kl \in |a|} w^a_{kl} \hat{F}^a_{kl}$$

where

$$w^a_{kl} = \frac{v^{kl}}{V_a} \tag{15}$$

and $|a|$ = the set of O-Ds that use aggregate link F^{IJ}_a (or ${}^K F^{IJ}_a$).

Finally, the portion of internal circulation mixed with line-haul traffic in an aggregate zone is to be assigned uniformly to all the access/egress and bypass links of the zone concerned. We denote this type of internal links I_K, where the subscript K denotes the aggregate zone in which the link can be found. The portion of internal circulation flowing on exclusive right-of-way (ROW) links, on the other hand, should be modeled as a separate intra-ROW link. We denote this type of internal links i_K. For example, shown in Table 1 is one internal trip from 4 to 6, using link (4, 6) in i_{II}. Likewise, the other internal trip from 6 to 4 uses link (6, 4) in I_{II}. The link aggregation procedure described above yields $I_{\text{II}} = 1½$ and $i_{\text{II}} = 1½$. For clarity, readers may wish to consult Figure 1 as they go through Table 1.

Aside from computational advantages, the aggregation procedure presented here has obvious functional advantages. The categorization of the detailed links in each aggregate zone into line-haul, access/egress, bypass, and internal circulation groups facilitates transportation analysis, as we can now conveniently refer to a generic class of detailed links by the particular function they perform (*15*). Thus a transportation planner can specify his/her improvement strategy in terms of the function performed by each aggregate link. For example, if egress from zone I to zone II is to be improved, $E^{\text{I II}}$ is specified as a candidate project.

While the example illustrates only single-path assignments, it is clear that the multipath assignment case represents a simple extension. Instead of weights w^a_{kl} defined for each O-D entry v^{kl}, it is now generalized to $w^a_{kl}(q)$, representing the O-D flows $v^{kl}(q)$ "fanning out" into the qth path. Thus the example can be carried forward without loss of generality (*18*).

AGGREGATE TREE SEARCH

Once a network is abstracted, the hierarchical search algorithm finds the optimal network design through a tree search procedure. A branch-and-bound algorithm will be used to search for the best link improvements to the abstracted network. To fix ideas, the algorithm here follows the classic tree search logic for binary variables, using the simple logic of "branching from the minimal lower bound." Other variants of the tree search—such as branch-and-backtrack—can be built upon the basic concepts here (*19*) and will be illustrated in sequel.

Bounding Rule

Every time a branch is made on the search tree, we evaluate the vehicle-minutes of travel resulting from improving a link or several links corresponding to the odd numbered node to the left and even numbered node to the right, respectively (see Figure 2). At each of these nodes one can write the following inequality (or bounds) for system-optimizing traffic assignments that are supposed to be performed:

$$E^c < E^n < E^o \tag{16}$$

where

E^o = vehicle-minutes of travel congestion before link shortening;
E^n = "arithmetic update" on travel congestion assuming flows do not shift paths after link improvement (hence only those that used the link benefit from travel-time reduction); and
E^c = travel congestion if there is a shift of flow paths.

If we use the same set of symbols, but underline them to denote the corresponding bounds in the aggregate search tree, we can write:

$$\underline{E}^c < \underline{E}^n < \underline{E}^o. \tag{17}$$

Now we will trace out the relationship between the $\underline{E}$'s in the aggregate network and the E's in the detailed network. Given that each aggregate network is derived on the basis of a fixed group of detailed links and some outdated detailed

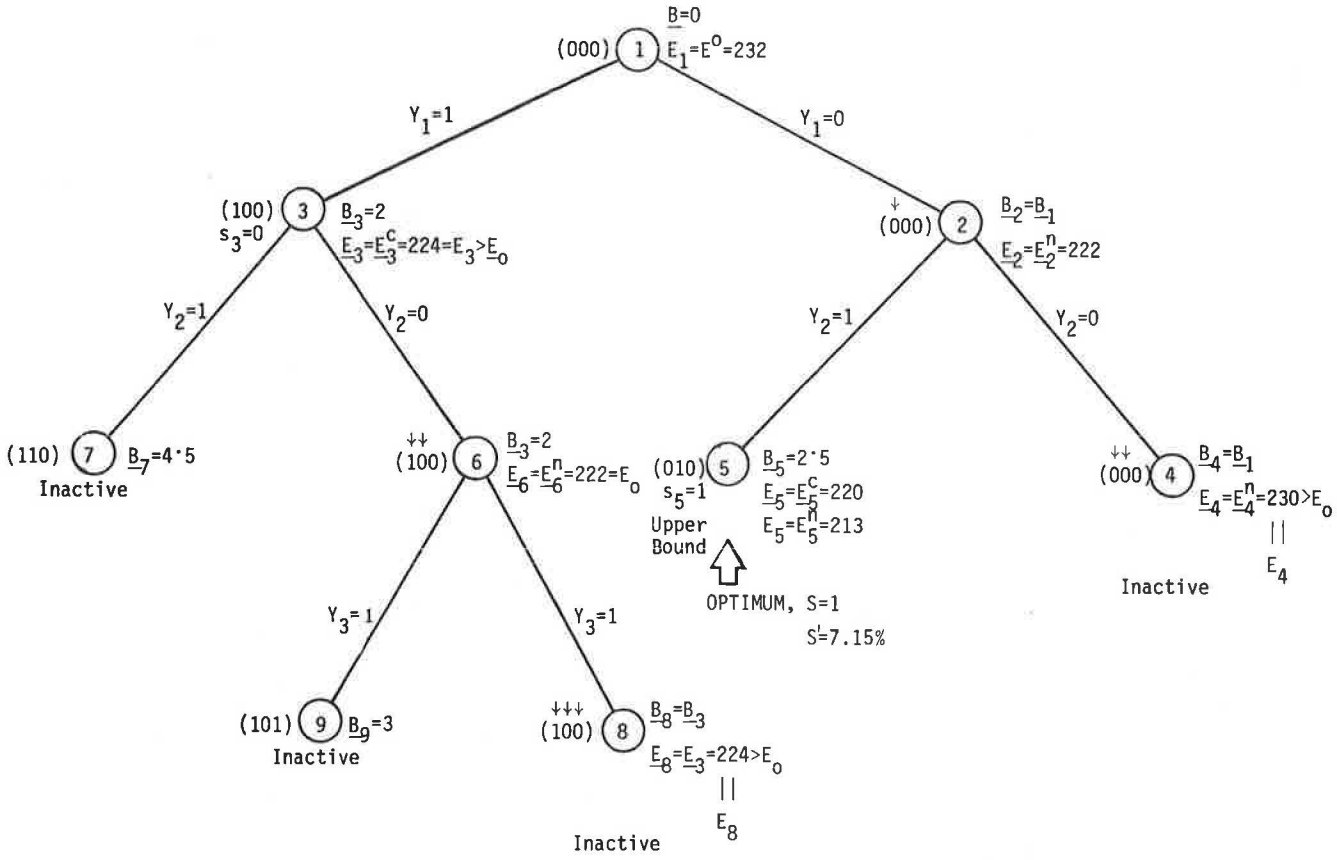

FIGURE 2 Aggregate branch-and-bound tree.

flow pattern further up the tree, one can show that at a particular node of the tree search (proof of this set of inequalities is shown in the Appendix):

$$\underline{E}^n \geq E^n \text{ and} \tag{18}$$

$$\underline{E}^c \geq E^n \tag{19}$$

where $\geq$ reads "is *likely to be* greater than or equal to."

Combining inequalities 16 through 19, the complete bounding relation can be written as

$$E^o = \underline{E}^o > \underline{E}^n > \underline{E}^c \geq E^n > E^c \tag{20}$$

Hence

$$\underline{E}_i \geq E_i \tag{21}$$

which says that, compared with the detailed assignment made at each node i of the aggregate tree, the aggregate system-optimizing travel congestion for the same node cannot be better. In other words, $\underline{E}$ is the upper bound for E.

Applying this bound to extend the "feasibility exclusion rule" of tree search, one can gain some efficiency at the bounded nodes by observing that as long as $\underline{E}^n$ is less than or equal to the maximally tolerable congestion E_o, we do not need to perform any traffic reassignment or "calibrate" the aggregate network against the detailed one. (*Calibration* is defined as performing a traffic assignment on the current detailed network and performing the network abstraction procedure again. More is said about calibration in the following section.) On the other hand, if $\underline{E}^n > E_o$ at node i, we examine the aggregate network and carry out postoptimality procedures to obtain $\underline{E}^c$. Either of the following cases may happen at a node i:

$$\underline{E}_i = \underline{E}_i^c \gtreqless E_o$$

1. If $>$, we calibrate by doing a detailed traffic assignment to obtain E_i.
2. If $\leq$, we keep on branching from this bounded node i.

Exclusion takes place only when the detailed reassignment indicates that $E_i > E_o$.

The exclusion criterion just described would guarantee a comprehensive search space in our aggregate tree search procedure, because we do not prune our tree by exclusion prematurely. A bounded node is excluded only if the detailed assignment indicates that no feasible solution can be obtained no matter what values the "free" variables pick up.

Equivalence Between Search in the Aggregate and Disaggregate Networks

The problem of disaggregation arises when one wants to translate network improvements in the aggregate network back to the detailed. Ideally speaking, one wishes to have the same network design as a detailed analysis, even though the analysis was actually performed in the aggregate search tree.

If, in the process of investigating congestion reduction in the aggregate search tree, we decided on shortening the travel

time on link F_a^{IJ} by $\Delta\hat{F}_a$, it gets translated to a corresponding set of detailed links $\{\Delta f_{ij}\}_a$:

$$\Delta\hat{F}_a \rightarrow \{\Delta f_{ij}\}_a \tag{22}$$

To evaluate Δf_{ij}s, we examine $\{F_{kl}^a\}$, the set of parallel chains defined in Equations 13 and 14. Because of our convex-combination definitions in aggregation, the amount of link shortening in each of the chain F_{kl}^a is the same:

$$\Delta\hat{F}_{kl}^a = \Delta\hat{F}_a \tag{23}$$

A series of link times $[f_{ij}]_{kl}^a$ may be contained in F_{kl}^a, according to Equation 14. This set of linear algebraic equations is therefore to be solved to obtain Δf_{ij}:

$$\sum_{ij \,\varepsilon\, |a_{kl}|} \Delta f_{ij} = \Delta\hat{F}_{kl}^a \qquad \forall a,\ \forall kl \tag{24}$$

It should be pointed out that the solutions of this set of equations are by no means unique. More often than not, they are indeterminate. Mathematical techniques alone are not able to resolve this problem satisfactorily. But irrespective of the arbitrary judgments made in solving Equation 24, the aggregate results of investment decisions are similar, as is shown below.

An essential part of making investment decisions in the aggregate space is to establish the equivalency between the aggregate and detailed networks. In other words, we wish to show the invariance properties both in the static networks and as we perform the search dynamically.

If the shortest paths do not shift, it can be shown that the total vehicle minutes of travel are conserved:

$$\begin{aligned}
\sum_a V_a\hat{F}_a &= \sum_a V_a \frac{\sum_{kl \,\varepsilon\, |a|} v^{kl}\hat{F}_{kl}^a}{V_a} && \text{via Equation 15}\\
&= \sum_a \sum_{kl \,\varepsilon\, |a|} v^{kl}\hat{F}_{kl}^a\\
&= \sum_a \sum_{kl \,\varepsilon\, |a|} v^{kl} \sum_{ij \,\varepsilon\, |a_{kl}|} f_{ij} && \text{via Equation 14}\\
&= \sum_a \sum_{kl \,\varepsilon\, |a|} \sum_{ij \,\varepsilon\, |a_{kl}|} v_{ij}^{kl} f_{ij}\\
&= \sum_{kl} \sum_{ij} v_{ij}^{kl} f_{ij} && (25)
\end{aligned}$$

A similar invariance relationship is maintained in project disaggregation. The equivalence of congestion reduction can be written as

$$\begin{aligned}
V_a\Delta\hat{F}_a &= \sum_{kl \,\varepsilon\, |a|} v^{kl}\,\Delta\hat{F}_a\\
&= \sum_{kl \,\varepsilon\, |a|} v^{kl}\,\Delta\hat{F}_{kl}^a && \text{via Equation 13}\\
&= \sum_{kl \,\varepsilon\, |a|} v^{kl} \sum_{ij \,\varepsilon\, |a_{kl}|} \Delta f_{ij}\\
&= \sum_{kl \,\varepsilon\, |a|} \sum_{ij \,\varepsilon\, |a_{kl}|} v_{ij}^{kl}\,\Delta f_{ij} && (26)
\end{aligned}$$

Although the discussions have been concentrating on disaggregation, the same invariance relationship can be developed for aggregation. Instead of specifying an aggregate link for improvement one can specify a set of detailed links, which are collapsed into the aggregate space prior to tree search.

While one can guarantee $E^o = \underline{E}^o$ by prohibiting path changes in the aggregate network, the invariance properties described above may not be guaranteed as one performs the tree search, which by its very nature causes path shifts. If too many path shifts occur, the weights used in network aggregation change according to Equation 15. This means that the original aggregate network may no longer be an accurate representation of the detailed. Under these circumstances, a calibration of the aggregate network is required, where a detailed traffic assignment is performed upon which a new abstract network is built.

The following heuristics should serve as guidelines for deciding when to calibrate at a node of the aggregate tree:

1. Using the project disaggregation rule, one can define the set of detailed links to be shortened and the amount for each aggregate investment project $\Delta\hat{F}_{IJ}$. Inspect (rather than actually compute) the routing matrices of the detailed network to estimate the number of detailed links that would cause flow shift; call this number e. (The postoptimality procedure of Murchland (*20*), for example, allows for a quick inspection of whether the shortening of a link introduces any flow shift.) If the flow shift is "significant," calibrate; otherwise, proceed with reassignment in the aggregate network.
2. Record the number of odd-numbered nodes in the aggregate tree that have been generated without calibration. This number, b, together with an estimate of the average number of detailed links that could cause path shifts at each odd-numbered node, e, gives a measure of the inaccuracy. If the inaccuracy exceeds a tolerance limit, calibrate; otherwise, proceed.
3. If the aggregate reassignment indicates a shift of flow but the detailed indicates otherwise, record the number of aggregate O-D flow path shifts, d. Together with b and e, d constitutes a measure of the inaccuracy of the aggregate tree search. Again, if the inaccuracy exceeds the tolerance limit, calibrate; otherwise, proceed.

To sum up the preceding three guidelines, we can define an error function at each node k of the aggregate tree as

$$s_k = be + d \tag{27}$$

The quality of a solution can be measured by the cumulative inaccuracies from the root node of the tree. The inaccuracy introduced by skipping calibration at a few odd-numbered nodes where detailed flows have been rerouted cannot be nullified by a calibration performed at the end of these "skips." We have to measure the inaccuracy of the solution by summing all the error functions at all calibration points skipped along the path, from the "root" of the tree to the solution at odd-numbered node K, where the quality of the solution, S, is to be measured.

$$S = \sum_{k \,\varepsilon\, (1 \rightarrow K)} s_k \tag{28}$$

One can normalize S by the total number of detailed links λ and the number of nonzero entries in the aggregate O-D matrix, P:

$$S' = \frac{S}{\lambda + P} \tag{29}$$

Such a percentage gives a rough estimate on the fraction of the total links and O-D pairs that have been affected by the path changes.

Following the same line of argument, the budget expenditure defined for an aggregate project changes as calibration takes place. The reason for this change is that the set of detailed links contained within the aggregate link changes over time as a consequence of path changes. Aggregate project costs, therefore, have to be redetermined at each calibration procedure by summing the costs of the current set of corresponding detailed projects. The point to be noted, however, is that this cost redefinition does not interfere with the additivity (hence, the monotonicity) of the budget function B_k.

Unfortunately, one cannot guarantee optimality in the aggregate search procedure—as the reader may have concluded already. Depending on how often we calibrate (i.e., regrouping the detailed links to a different aggregate link), the result of the optimization routine would conceivably be different. Even when calibration is performed at each node, the final detailed project selection would still depend on the order in which the $0 - 1$ decision variables Y_i are introduced into the tree. The reason for that relationship is that the disaggregation of aggregate link improvement to the detailed level would depend on the current grouping of detailed links to aggregate links, and the grouping is again a function of the order in which Y_i's are introduced.

Branch-and-Bound Algorithm

We are now ready to formalize the tree-search algorithm.

Step 0. Perform network aggregation at the root node $r = 1$. Define for this node $\overline{Y} = (\overline{0})$, and label it with objectives function $\underline{B} = \underline{B}_1 = 0$.

Step 1. *Bound*: Out of the set of active nodes, find the node l with the smallest objective function $\underline{B}_l$ (i.e., the lower bound T). Node l is the bounded node. If $r \neq 1$, set $r = r + 2$.

Step 2. If an active node j has $\underline{E}_j \leq \underline{E}_o$, a reassignment is performed at the detailed level, yielding a new system cost E_j. Compute the error function s_j, which indicates the need for calibration. Then set upper bound $U = \underline{B}_j$. Put node j on an inactive status. All active, feasible nodes with $\underline{B}_i \geq U$ are dominated and declared inactive. If there are no more active nodes, terminate the algorithm. The optimal solution $\underline{B}_j^* = U$ 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 $Y_k = 0$ on the right branch and $Y_k = 1$ on the left branch. At node $r + 1$, an arithmetic update is performed, with the free variables set to 1. If $\underline{E}_{r+1} > \underline{E}_o$, obtain E_{r+1}. If $E_{r+1} > \underline{E}_o$, node $r + 1$ has been fathomed and termed inactive. Otherwise, set $\underline{B}_r = \underline{B}_l$. At node $r + 2$, compute $\underline{B}_{r+2} = \overline{\beta}^T \overline{Y}_{r+2}$. Go back to Step 1.

NUMERICAL EXAMPLE

Take the base network shown in Figure 1, which has a total congestion cost of E^o 232 vehicle-minutes in both the aggregate and detailed networks. We will define three aggregate projects, which improve the line-haul, access, and bypass functions rendered by the network, respectively:

Project 1—shorten $L^{\mathrm{I\,II}}$	by $\Delta\hat{L}^{\mathrm{I\,II}}$	= 2 minutes
Project 2—shorten $A^{\mathrm{I\,II}}$	by $\Delta\hat{A}^{\mathrm{I\,II}}$	= 4 minutes, and
Project 3—shorten ${}^{\mathrm{II}}B^{\mathrm{I\,III}}$	by $\Delta^{\mathrm{II}}\hat{B}^{\mathrm{I\,III}}$	= 1 minute.

The costs for Projects 1, 2, and 3 are estimated from their detailed counterparts as 2, 5, and 1 units, respectively. The objective of tree search is to find the lowest-budget network improvement plan, with the tolerable congestion level E_o set at 222 vehicle-minutes.

In Figure 2, we perform the branch-and-bound tree-search algorithm step by step, as illustrated:

- *Initialization*: At node 1, $\overline{Y}_1 = (0,0,0)$ at budget level $\underline{B} = 0$, and $\underline{E}_1 = E^o = 232$. We branch to nodes 2 and 3, corresponding to $Y_1 = 0$ and 1, respectively.
- *Nodes 2 and 3*: Node 2 is the bounded node, with a lower budget $\underline{B}_2 = \underline{B}_1 = 0$ (compared with a $\underline{B}_3$ of 2 at node 3). At node 2, variable Y_1 is fixed at zero value, leaving only Y_2 and Y_3 free. An arithmetic update is performed at node 2, with the free variable set to 1: $\underline{E}_2 = \underline{E}_1 - V_2\Delta\hat{F}_2 - V_3\Delta\hat{F}_3 = 232 - 2 \times 4 - 2 \times 1 = 222 = \underline{E}_2^n$. We branch from node 2 to nodes 4 and 5.
- *Nodes 4 and 5*: At the bounded node 4, an arithmetic update is performed: $\underline{E}_4^n = \underline{E}_1 - V_3\Delta\hat{F}_3 = 232 - 2 \times 1 = 230 > E_o$. According to the logic described in the bounding rule portion of the aggregate tree search section, the true state of the system has to be assessed. A reassignment is made on the aggregate network, resulting in $\underline{E}_4 = \underline{E}_4^n$. This prompts another reassignment on the detailed network corresponding to $\Delta f_{5\,4} = 1$, resulting in $E_4 = \underline{E}_4 > E_o$. We exclude further branching from node 4. Between terminal nodes 3 and 5, node 3 is the bounded node because it carries a lower B of 2. A reassignment at node 3 yields $\underline{E}_3 = 224 = E_3$, with an error function $s_3 = b \cdot e = 1 \times 0 = 0$. We branch to nodes 6 and 7.
- *Nodes 6 and 7*: Between 5, 6, and 7, node 6 is the bounded node. An arithmetic update yields $E_6^n = E_3 - V_3\Delta F_3 = 224 - 2 = 222 = E_o$. Further branching is to be performed from this node.
- *Nodes 8 and 9*: Bounded node 8 carries $\underline{E}_8 = 224 = \underline{E}_3 > E_o$. We exclude further branching from this node after checking that $E_8 = \underline{E}_8 = 224 > E_o$.

Out of the three terminal nodes (5, 7 and 9), node 5 is the bounded one because it carries the least budget. The reassigned system cost $\underline{E}_5 = \underline{E}_5^c = 220$. The flows that formerly traversed ${}^{\mathrm{II}}B^{\mathrm{I\,III}}$ are now rerouted using $A^{\mathrm{I\,II}}$ and $E^{\mathrm{II\,III}}$ as bypass links. In the detailed network $E_5 = E_5^n = 213$. The error function $s_5 = b \cdot e + d = 1 \times 0 + 1 = 1$.

- *Termination*: Node 5 is a feasible node with a budget objective function of 2.5; $B_5 = 2.5$ would be an upper bound to help reject further branching from any nodes with B's higher than 2.5. By this rule, nodes 7 and 9 are rejected. The optimum of improving $\hat{L}^{\mathrm{I\,II}}$ [or link (3,5)] by 2 min is found. The error function measures

$$S = \sum_{t \in (1 \to 5)} s_t = 1$$

or a percentage error of $S' = 1/(10 + 4) = 7.15$ percent.

CASE STUDY

We further illustrate the hierarchical search algorithm through a case study of the 1978 network from Taipei metropolitan area, Taiwan, Republic of China, consisting of 49 zones (see Figure 3). To show the versatility of aggregate tree search, the following network design formulation, rather than the one used in the numerical example, is used:

$$\text{minimize } E(\bar{y}) = \min \sum_{kl} \sum_{ij \in R^{kl}} (f_{ij} - \Delta f_{ij} y_{ij})\, x_{ij}^{kl} \tag{30}$$

subject to the node-arc incidence relationship of Equation 5 and the budget constraint

$$\bar{\beta}^T \bar{y} \leq B \tag{31}$$

Essentially, this is the inverse of the formulation in Equations 1 through 6, wherein the best congestion reduction is to be achieved within the budget allowance. In Equation 30, the travel time function f_{ij} assumes the form

$$f_{ij} = f_{ij}^o \left(\frac{1 - h\rho_{ij}}{1 - \rho_{ij}} \right) \tag{32}$$

where ρ_{ij} is the volume/capacity ratio or

$$\rho_{ij} = \frac{x_{ij}}{c_{ij}} \tag{33}$$

Typical capacities (c) range from 1,250 to 3,400 passenger-car-equivalents per hour in the study area, covering local streets through superhighways; h is a calibration constant with typical values of .4, .5, and .6.

The network aggregation algorithm reduces the number of zones from 49 to 14 (see Figure 3), nodes from 155 to 76, and links from 568 to 222. It follows exactly the same procedure as the previous example except that a multipath assignment is used as described earlier.

Three line-haul link-improvement projects were identified.

Project 1: $\Delta\hat{L}^{\text{VII XII}} = 5$ min at a cost of 2 units,

Project 2: $\Delta\hat{L}^{\text{XIV V}} = 5\frac{1}{2}$ min at a cost of 1.5 units, and

Project 3: $\Delta\hat{L}^{\text{XIV I}} = 8\frac{1}{2}$ min at a cost of 2.5 units.

A total budget of four units is imposed.

Branch-and-backtrack is used in the tree search rather than branch-and-bound. For simplicity, the tree search was con-

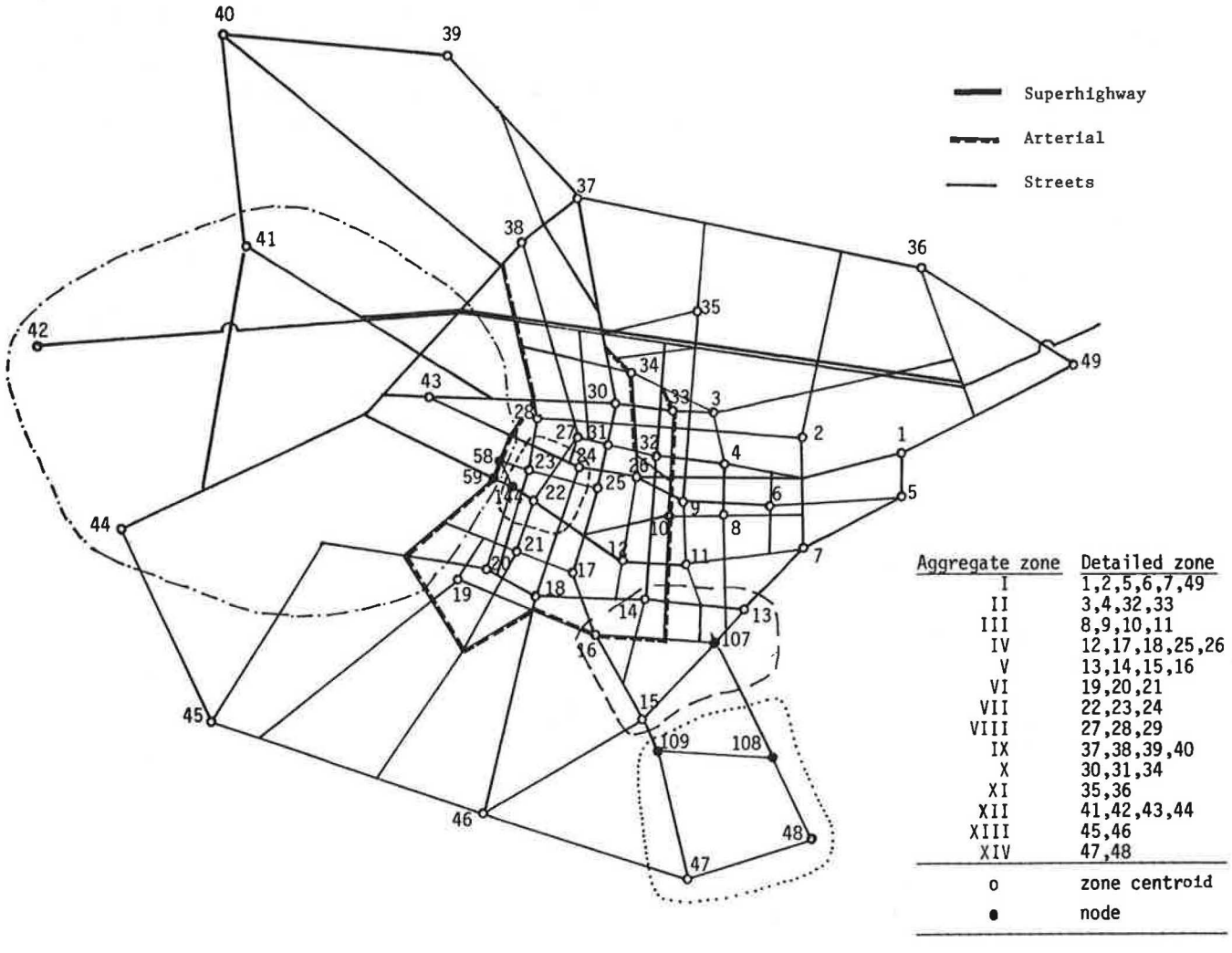

FIGURE 3 Taipei metropolitan network.

ducted entirely in the aggregate network without calibration. Instead of branching from the lowest bound, the branch-and-backtrack method keeps on branching from the latest active node (*19*):

- $r = 1$: Initialize at $\overline{Y}_1 = (0,0,0)$, with $E_1 = 6.6919 \times 10^6$ vehicle-minutes. Branch to nodes 2 and 3, corresponding to $Y_1 = 1$ and 0, respectively. Because $\underline{B}_2 = b_1 = 2 < B$, node 2 is declared active. On the other hand, as $\underline{B}_3 = b_2 + b_3 = 4 \leq B$, node 3 is a feasible solution. The reassignment performed at node 3 yields $\underline{E}_3 = 6.5112 \times 10^6$ vehicle-minutes, which is a better level of system congestion than $\underline{E}_1$. Set $U = \underline{E}_3$ and declare node 3 as inactive. Branch from node 2.
- $r = 3$: Nodes 4 and 5 are obtained corresponding to $Y_2 = 1$ and 0. Because $\underline{B}_4 = b_1 + b_2 = 3.5 < B$, node 4 is active. Similarly, node 5 is also active because $\underline{B}_5 = b_1 + b_3 = 4.5 > B$.
- $r = 5$: Branch from node 5 to nodes 6 and 7. Because $\underline{B}_6 = b_1 + b_3 = 4.5 > B$, declare node 6 to be inactive. On the other hand, $\underline{B}_7 = b_1 = 2 < B$, node 7 is feasible. Traffic assignment shows that $\underline{E}_7 = 6.6386 \times 10^6 > U$. Node 7 is declared inactive; node 4 is the only place where branching can take place.
- $r = 7$: Branch from node 4 to 8 and 9. As $\underline{B}_8 = b_1 + b_2 + b_3 = 6 > B$, node 8 is declared inactive. $\underline{B}_9 = b_1 + b_2 = 4 \leq B$ and $\underline{E}_9 = U = 6.5712 \times 10^6 = \underline{E}^*$. We stop branching because the optimum has been found, corresponding to improving $L^{\text{VII XII}}$ and $L^{\text{XIV V}}$.

For actual implementation purposes, we disaggregate $L^{\text{VII XII}}$ into the parallel chains of (144, 59) and (23, 58) and $L^{\text{XIV V}}$, into parallel chains of (108, 107) and (109, 15). From Equation 14:

$$\Delta\hat{F}^{L}_{22\ 44} = \Delta f_{144\ 59} = 5, \text{ and}$$

$$\Delta\hat{F}^{L}_{23\ 43} = \Delta f_{23\ 58} = 5;$$

yielding the obvious answer of a travel time reduction of 5 min for both detailed links (144,59) and (23,58). Likewise,

$$\Delta\hat{F}^{L}_{48\ 13} = \Delta f_{108\ 107} = 5.5 \text{ and}$$

$$\Delta\hat{F}^{L}_{47\ 15} = \Delta f_{109\ 15} = 5.5;$$

which means a reduction of 5.5 min for both (108,107) and (109,15).

This case study illustrates the practical value of the hierarchical search algorithm in solving realistic size network problems. The city of Taipei, with its population of 2.4 million, was analyzed using a FORTRAN IV traffic assignment code on IBM OS/360. In spite of the use of this relatively outdated machine, the multipath assignment in the detailed network took less than 5 min to cover all the computational trials, yielding an adopted h value of 0.6 in Equation 32. The network design tree search on the aggregate network was also simple enough to be conducted by hand as already shown. It is estimated that an eightfold savings of computational requirement was achieved. Most of this is the result of network abstraction, which reduces the number of links/nodes, hence expediting traffic assignment and tree search in an exponential manner. Because no calibration was performed, however, there is no measurement on the quality of the approximate solution.

COMPARISON OF NETWORK ABSTRACTION WITH NETWORK EXTRACTION

To assess further the afore-described abstraction algorithm in network design applications, a comparison is made with the previously mentioned network extraction algorithm of Haghani and Daskin (*3*). A common square-grid network of 25 nodes and 40 links as shown in Chan (*5*) is analyzed, in which a kth best path traffic assignment (*21*) is performed.

In this controlled experiment, the second network design formulation (Equations 30 and 31) is used, wherein the budget constraint allows for the implementation of only one of the two candidate projects. The first project reduces the cost of two detailed links from 15 to 10 and from 10 to 5 min of travel time, respectively. The second project reduces one single detailed link by three separate amounts: 15 to 9 (Case A), 15 to 8 (Case B), and 15 to 5 (Case C). Results of this experiment are shown in Table 2.

The results show that investment decisions made on both the abstracted network and the extracted network agree to a large extent with those on the detailed network. Two of the three link improvement decisions are the same between the detailed analysis and each aggregate algorithm. The abstraction algorithm performs better in estimating both the total congestion (in vehicle-minutes) and the congestion reduction of Project 1. The extraction algorithm, on the other hand, yields identical congestion reduction as in the detailed analysis for Project 2.

Although this constitutes only a limited experimentation, a few observations can be made.

1. Because of the "invariance" property of abstraction, traffic assignment in the abstracted network appears to yield a total system congestion closer to the detailed network than to the extracted network.
2. Depending on the candidate links for network improvement, the congestion reduction effect may be estimated to be different by the two aggregation schemes. In the case of a

TABLE 2 ABSTRACTION AND EXTRACTION COMPARED

	Detailed Network	Abstracted Network	Extracted Network
Total congestion (vehicle-min)	2,385[a]	2,178[a]	2,125
Congestion reduction for project 1 (vehicle-min)	145	134	130
Congestion reduction for project 2 (vehicle-min)			
Case A	138	94	138
Case B	161	109	161
Case C	230	157	230
Selected project (vehicle-min)			
Case A	1	1	2
Case B	2	1	2
Case C	2	2	2

[a]The two figures are not the same due to path shifts in the abstracted network. By definition, figures are identical without reassignment.

relatively insignificant candidate link, abstraction yields better results. For a major link, on the other hand, extraction algorithm works better. This shows that abstraction is a more "balanced" aggregation algorithm in which the properties of all links—both major and minor—are included in the aggregate network, whereas extraction tends to favor major links to the minor ones, inasmuch as the algorithm explicitly retains major links and discards minor ones.

3. Owing to their different premises, one should not expect the abstraction algorithm to yield identical results as extraction, although there should be some similarity between analyses performed on the aggregated networks and the detailed networks, irrespective of the aggregation method.

CONCLUSION

In this paper, we applied a hierarchical search algorithm to solve network design problems for three spatially abstracted networks. The branch-and-bound and branch-and-backtrack techniques, respectively, were used in the first two formulations of the problem, assuming the objective function of least budget and least travel cost, respectively. These techniques result in a greatly reduced search space, as well as a functional grouping of the detailed links into access/egress, line-haul, and bypass categories. The latter allows network improvement projects to be specified in terms of the corresponding aggregate links that are identified by the function they perform.

The hierarchical search algorithm—combining network abstraction with tree search—was also shown to possess tighter bounding criteria than tree search alone, thus accelerating computational efficiency. Equivalency was established between the aggregate space and the detailed space, including certain invariance properties, such as conservation of vehicle-minutes of travel between the abstracted and detailed networks. The inaccuracy introduced by the approximate optimization procedure was measured by an "error function," showing the solution's percentage divergence from the global optimum. Aside from a numerical example, a case study was taken from the metropolitan area of Taipei, Taiwan, Republic of China, to illustrate the usefulness of the algorithm. For example, the case study shows that computational requirement is reduced by a factor of 8 (*22*).

To the writers' best knowledge, and in their opinion, the hierarchical search algorithm is the first "scientific" attempt to establish equivalency between abstracted and detailed network design decisions. The only required system behavior is monotonicity of two figures of merit as the tree search is being conducted; in our formulation, these are cumulative project expenditure and system congestion level. As such, it is an extremely flexible technique to take care of this class of NP-hard problems. For example, it is conceivable that elastic demands can be tackled as long as monotonicity is maintained in the figures of merit chosen.

The abstracted network is, in essence, a convex combination of the link and node attributes of the original network. This convexity property is exploited to the fullest in establishing the equivalency already mentioned, including the conservation of system travel between the abstracted and detailed networks during aggregation and disaggregation. Should no path shift at all during the search, strict equivalency is guaranteed. The network design methodology becomes approximate when paths do shift as a result of reassignment or network improvement, and the divergence from global optimum in this case is then measured by an error function (in percentages), as alluded to earlier.

The abstraction method was compared with extraction on the third and last network in our research. It was found that both yield investment strategies approximating the detailed network design model, although there is little correlation between the two aggregation approaches. Because of its invariance property, abstraction appears to estimate the total system congestion (in vehicle-minutes) more accurately. There is little distinction between the two, however, in estimating the specific congestion-reduction associated with link improvements.

Although the preceding findings represent modest progress, much work remains to be done in the hierarchical search algorithm. Among them are the following:

1. Further computational experience can be gained by relating the "error function" to the sequence with which link improvement projects are introduced into the search tree and the frequency of calibration. The objective is to find a strategy that minimizes the error in aggregate search.
2. Another set of experiments can be performed to clarify the trade-off between the level of abstraction and the inaccuracy introduced into the solution. The objective is to know the "appropriate" level of aggregation for a given problem.
3. "Branching from the minimal lower bound or the latest active node" is used throughout this paper. Although it served to introduce the hierarchical search algorithm, more efficient tree searches along the line of work by Chan (*19*) can be used to gain even better computational efficiency through the use of double bounds.
4. In spite of the insights gained in our experiments, additional tests can obviously be conducted to compare the performance of an abstraction approach with the extraction approach in network design.

It will be useful eventually to generalize the algorithm to perform user-optimizing network designs, in addition to the system-optimizing ones performed here—although this is by no means a straightforward extension inasmuch as monotonicity properties are no longer guaranteed in the tree search due to Braess's Paradox.

ACKNOWLEDGMENTS

The authors acknowledge the support of the U.S. Department of Transportation, U.S. Department of Defense, General Motors, and the Chinese Technical-Service Council. They are grateful to the many colleagues and friends, including M. L. Manheim, J. H. Stafford, T. L. Loung, and many others at the Air Force Institute of Technology, Massachusetts Institute of Technology, and National Taiwan University, who were of aid. T. S. Shen's contribution toward this paper was performed while he was working in Tapei, Taiwan (*22*). Naturally, the individual authors, rather than their affiliations, are solely responsible for the statements made here.

APPENDIX: Proof that the System Cost in an Aggregate Search, $\underline{E}$, is the Upper Bound for the Corresponding Statistic, E, in the Detailed Space

There are two possible consequences of shortening an aggregate link at node k of the tree search: either a path shift occurs or it does not—that is

$$\underline{E}_k = \underline{E}_k^c \text{ or}$$

$$\underline{E}_k = \underline{E}_k^n.$$

To show the inequality between $\underline{E}_k$ and E_k, one has four possible pairwise relationships to consider.

1. $\underline{E}^n - E^n$;
2. $\underline{E}^n - E^c$;
3. $\underline{E}^c - E^n$; and
4. $\underline{E}^c - E^c$.

Our objective is to show that in all four cases, $\underline{E} \geq E$.

Case 1. When there are no path shifts, shortening the aggregate link will, by virtue of Equation 26, yield the same vehicle-minute reduction as shortening the corresponding set of detailed links; hence $\underline{E}^n = E^n$.

Case 2. This follows from the result of the first case. Because $\underline{E}^n = E^n$ and $E^n > E^c$ from Equation 16, it follows that $\underline{E}^n > E^c$.

Case 3. A detailed link may get categorized under more than one aggregate link, as can be seen in Equation 13 and in Table 1 for $f_{6\ 4}$. Disaggregation of an aggregate link improvement into the detailed network at a node of the aggregate tree means that each improved detailed link (i, j) benefits diverse O-Ds (including intra flows) that use (i, j) according to Equation 30, even though no path shifts occur by definition. For the aggregate link, on the other hand, functional restrictions are placed in the flow paths given that each aggregate link is derived on the basis of some outdated flow pattern further up the tree following Equations 13 through 15. Even though paths shift in the abstracted network owing to the link improvement, the limited amount of flexibility in the outdated network does not allow as great a vehicle-minutes of travel reduction as the corresponding arithmetic update in the detailed network where each and every unit of reduction is accounted for. This means that the aggregate travel congestion is likely to be greater than or equal to that of the detailed after link improvement as illustrated in node 5 of Figure 2.

$$\underline{E}^c \geq E^n.$$

Case 4. This case follows when one combines the result of the preceding finding with Equation 16 (or $E^n > E^c$). Therefore

$$\underline{E}^c > E^c.$$

Conclusion: In all cases, $\underline{E} \geq E$.

REFERENCES

1. R. Wong. Worst-Case Analysis of Network Design Problem Heuristics. *SIAM Journal*, Vol. 1, No. 1, 1980.
2. Y. Chan, K. Follansbee, M. Manheim, and J. Mumford. Aggregation in Transportation Networks: An Application of Hierarchical Structure. Research Report R68-47. Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, 1968.
3. A. Haghani and M. Daskin. Network-Design Application of an Extraction Algorithm for Network Aggregation. In *Transportation Research Record 944*, TRB, National Research Council, Washington, D.C., 1984, pp. 37–60.
4. P. Bovy and G. Jansen. Network Aggregation Effects upon Equilibrium Assignment Outcomes: An Empirical Investigation. *Transportation Science*, Vol. 17, No. 3, 1983, pp. 240–261.
5. Y. Chan. A Method to Simplify Network Representation in Transportation Planning. *Transportation Research*, Vol. 10, 1976, pp. 179–191.
6. R. Eash, K. Chou, Y. Lee, and D. Boyce. Equilibrium Traffic Assignment on an Aggregated Highway Network for Sketch Planning. In *Transportation Research Record 944*, TRB, National Research Council, Washington, D.C., 1984, pp. 30–37.
7. P. Zipkin. Bounds for Aggregating Nodes in Network Problems. *Mathematical Programming*, Vol. 19, 1980, pp. 155–177.
8. J. R. Evans. The Multi-Commodity Assignment Problem—A Network Aggregation Heuristic. *Computers and Mathematics with Applications*, Vol. 7, No. 2, 1981, pp. 187–194.
9. M. Abdulaal and L. LeBlanc. Continuous Equilibrium Network Design Models. *Transportation Research*, Vol. 13B, 1979, pp. 19–32.
10. T. Friesz. Transportation Network Equilibrium, Design and Aggregation: Key Developments and Research Opportunities. *Transportation Research A*, Vol. 19A, No. 516, 1985, pp. 413–427.
11. C. Suwansirikul, T. Friesz, and R. Tobin. Equilibrium Decomposed Optimization: A Heuristic for the Continuous Equilibrium Network-Design Problem. *Transportation Science*, Vol. 21, No. 4, 1987.
12. R. Barton and D. Hearn. Network Aggregation in Transportation Planning. U.S. Department of Transportation, Mathematica, Princeton, N.J., 1979.
13. Y. Chan. Optimal Travel Time Reduction in a Transport Network: An Application of Network Aggregation and Branch-and-Bound Techniques. Research Report R69-39. Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, (Clearinghouse AD 693-095), 1969.
14. G. Dantzig, S. Maier, and Z. Lansdowne. The Application of Decomposition to Transportation Network Analysis. Report DOT-TSC-OST-76-26. U.S. Department of Transportation, Washington, D.C., 1978.
15. Y. Chan. A Comparative Analysis of Network Aggregation Approaches. Presented at the National Joint Meeting of the Operations Research Society of America and the Institute of Management Science, San Francisco, 1977.
16. F. Ou and Y. Chan. An Aggregate Representation of Areawide Urban Networks. *Proc., 11th Annual Pittsburgh Conference on Modelling and Simulation*, University of Pittsburgh, 1980, pp. 1115–1119.
17. Y. Chan and F. Ou. Equilibration Procedure to Forecast Areawide Travel. *Transportation Engineering*, Vol. 112, No. 6, 1986, pp. 557–575.
18. N. M. Mahaba. *Network Aggregation*. Paper for OPER 7.67—Network and Combinatorial Optimization, Department of Operational Sciences, Air Force Institute of Technology, Wright-Patterson, Ohio, 1988.
19. Y. Chan. The Transportation Network Investment Problem: A Synthesis of Tree-Search Solution Algorithms. *Transportation Research Record 1074*, TRB, National Research Council, Washington, D.C., 1986, pp. 32–40.
20. J. D. Murchland. A Fixed Matrix Method for All Shortest Distances in a Directed Graph and for the Inverse Problem. Doctoral thesis. University of Karlsruhe, West Germany, 1970.
21. M. Pollack. Solutions of the kth Best Route—A Review. *Journal of Mathematical Analysis and Applications*, Vol. 3, 1961, p. 547.
22. T. S. Shen. Application of Traffic Assignment and Network Aggregation in an Urban Area. Master's thesis. National Taiwan University, Taipei, Taiwan, Republic of China, 1981.