Three heuristics for improving centralized routing in large long-haul computer communication networks

by IVAN M. PESIC and DANIEL W. LEWIS
University of Santa Clara
Santa Clara, California

ABSTRACT

Several algorithms have proven to be useful in computing the shortest path between two nodes in a network. Their complexity depends on the problem definition and size of the network measured by the total number of nodes N. In these algorithms, finding all the shortest paths from a fixed node is a computation of complexity $O(N \times N)$. These algorithms have been used successfully in long-haul networks for many years. Recent growth of such networks to large numbers of nodes (e.g., $N > 500$) demands a more efficient approach. This paper develops three heuristics based on measured topological characteristics of computer communication networks and applies them in the construction of faster algorithms. Determination of the shortest path between two random nodes is then shown to require 50 to 60% less computation.
INTRODUCTION

Background

The problem of finding the shortest path in a given network is not a new one. Dijkstra simply stated the problem as:

We consider \( N \) points (nodes), some or all pairs of which are connected by a branch; the length of each branch is given. We restrict ourselves to the case where at least one path exists between two nodes. Find the path of minimum total length between two given nodes \( P \) and \( Q \).

In the late 1950s, Kruskal\(^1\) and Dijkstra\(^2\) proposed the first algorithmic solutions to the routing problem. These and subsequent algorithms\(^3-7\) all have satisfactory performance when applied to a network with a small number of nodes, but as the network grows larger, the number of necessary operations grows polynomially with \( N \). Such algorithms are simply not fast enough for routing applications in large networks (e.g., \( N > 500 \)). Hierarchical routing has been suggested as one solution; particular applications for networks with distributed control have appeared in the literature.\(^9,10\)

Dijkstra's problem definition is intentionally general and thus simplistic. If taken literally, it assumes a connected topology with otherwise random characteristics, providing only an assigned cost for each pair of nodes (with a cost of infinity signifying no connection) to guide the routing algorithm. Such a problem definition anticipates absolute minimal cost solutions as the goal, and is not necessarily appropriate for applications in which routing time is important. In such a case, a good solution is often better if it can be computed in significantly less time. This may be achieved by restricting the algorithm's search of the topology through heuristics which use additional information about the environment to avoid useless avenues of exploration.

This paper defines some additional characteristics of computer communications networks that expand the information over that provided in the original problem statement, and then uses this information in the application of heuristics to the routing problem. It has been argued\(^11\) that only non-adaptive (or semi-adaptive) routing will be effective in the future environment of very large networks due to the enormous overhead that would be required for a fully adaptive approach. Non-adaptive routing is traditionally associated with centralized routing; therefore this paper addresses only this type. The results are from a study of the topology of real and simulated computer communication networks and the final topology of their routed paths.

Definitions

Long-Haul Network: Computer communications networks in which the physical distribution of resources and users may span large geographical areas, such as the entire United States. This paper is primarily concerned with large long-haul networks (e.g., \( N > 500 \)) in which centralized routing suffers from computational complexity.

Link: A bidirectional communication path in a computer network with distinct ends.

Node: A terminus of network links, usually a traffic switching center, a computation center, or both.

Chain: A finite sequence of two or more links in which each pair of links within the sequence is connected by a single node of degree two.

Link cost: A real number \( c(l) \) that is assigned to each link \( l \) in a computer network, periodically adjusted in value to control the distribution of traffic. Different costs may be assigned to each direction associated with a link for total generality; the results of this work may be applied in either case.

Connectivity (or degree) of a node: The number of links emanating from it to connect it with other nodes.

Characteristics of Long-Haul Networks

Large computer communication networks possess several very important characteristics that can be used to develop heuristics to accelerate the routing algorithms. This section represents a study of the inherent characteristics of the TYMNET computer network as it has matured through three developmental stages. These characteristics are representative of most long-haul networks, and the derived heuristics are universally applicable.

Geographical

When a long-haul network is relatively small (e.g., \( N < 50 \)), most of its nodes will be found in large cities. Using the United States as an example, nodes would be found near population centers along the east and west coasts, in the south, and in the north-central area (e.g., Chicago), with long communication lines spanning the large central region. These first nodes serve as initial computational centers, around which large local networks develop as the network expands. As computational loads increase, more nodes are added in other cities. Once the network reaches about 200 nodes, the nodes will be fairly evenly distributed across all the states.
Connectivity

Distributions of connectivity for three different stages of the development of the TYMNET computer network are shown in Figure 1. Small networks are relatively loosely connected, and network growth is followed by a limited growth of average connectivity. Correspondingly, nodes of degree one were predominant when TYMNET was still small; as the network expanded, nodes of degree two became predominant. This growth is reflected by average connectivities of 2.4, 3.4, and 3.9 for 1971, 1975, and 1977, respectively. Taken together, nodes of degree one and two now comprise about one third of the total number of nodes.

Link costs

Network performance (e.g., response time) is very sensitive to traffic distribution that is directly controlled by the assignment of link costs (or lengths, as defined in Dijkstra's problem statement). In particular, the routing algorithm and cost assignment should focus on the following objectives:

1. Avoid congestion and balance the load across the network.
2. Minimize the response time or delay for interactive users.
3. Provide adequate bandwidth for high-speed users.
4. Maximize the resource utilization.

The actual assignment of link costs in TYMNET, shown in Table I, was used in this study as a basis on which to synthesize a model of the TYMNET network, with variations in the total number of nodes and average connectivity.

Time variance

Computer communication networks are time-varying systems, and as such their parameters must be occasionally adjusted to accurately reflect changing conditions. For example, the assignment of link costs is adjusted periodically to control the distribution of traffic within the network, typically according to the time of day or day of the week. When the network is heavily loaded, utilization of individual nodes and links can change quickly; link costs must then be more frequently reassigned in order to maintain an equitable distribution of the workload, and thus a reasonable level of performance. For example, in the TYMNET network, the costs

<table>
<thead>
<tr>
<th>Line Type</th>
<th>Cost (arbitrary units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9600 bps</td>
<td>10</td>
</tr>
<tr>
<td>7200 bps</td>
<td>11</td>
</tr>
<tr>
<td>4800 bps</td>
<td>12</td>
</tr>
<tr>
<td>2400 bps</td>
<td>16</td>
</tr>
</tbody>
</table>

Figure 1—Connectivity distribution for the TYMNET network (a) 1971, (b) 1975, and (c) 1977
are usually reassigned once every few minutes, but sometimes as often as every few seconds during periods of high load, such as around noon.

Actual network topology changes less frequently, usually owing to the addition or deletion of nodes and links as part of normal growth. A typical network can grow as fast as a node per week. Long-term failure of a node or link may, of course, require a change in topology, but more often this is handled as a temporary assignment of infinity cost (no connection) to the link (or links) involved.

Network Synthesis for Simulation

For the construction of a network model for simulation purposes, it can be safely assumed that small networks have average connectivities in the range 2 to 3, medium networks in the range 3 to 4, and large networks greater than 4. In order to evaluate the new routing heuristics, a computer program was written to model long-haul networks with varied characteristics. The program consists of four parts:

1. Part one is responsible for placing a desired number of nodes $N$ over a rectangular region representing the United States. The nodal coordinates are generated using a random number generator with a uniform probability, and a period longer than 50,000 numbers.

2. After nodal placement, the second part of the program assigns a connectivity degree to each node. The user supplies the desired average connectivity and maximal allowed nodal degree for the network to be constructed. The program first computes the number of links to be placed using the formula

$$\text{number of links } L = \frac{(\text{number of nodes} \times \text{average connectivity})}{2},$$

and then solves a system of equations:

$$\sum_{i=1}^{\text{max}} N_i = N, \quad (1)$$

where $N_i$ is the number of nodes of degree $i$,

$$\left[ \sum_{i=1}^{\text{max}} N_i \times i \right]/2 = L, \quad (2)$$

and

$$N_i = \left[ c_1 \exp\left(-\left(i - 2\right)^2/c_2^2\right) \right], \quad (3)$$

where $200 < N \leq 300$, $i_{\text{max}} = 9$; $100 < N \leq 200$, $i_{\text{max}} = 6$; $0 < N \leq 100$, $i_{\text{max}} = 3$;

$$N_i = \left[ \frac{N - \sum_{i=1}^{\text{max}} N_i}{(\text{max nodal degree} = i_{\text{max}})} \right], \quad (4)$$

where $200 < N \leq 300$, $i_{\text{max}} = 9$; $100 < N \leq 200$, $i_{\text{max}} = 6$; $0 < N \leq 100$, $i_{\text{max}} = 3$;

The $i_{\text{max}}$ nodal degree $\geq k \geq i_{\text{max}}$. Equation (3) models the distribution of nodes of small degree (exponentially, with constants $c_1$ and $c_2$ supplied by the user) while equation (4) models the distribution of nodes of large degree. The $N(i)$ versus $i$ function obtained by solving equations (1)–(4) is then used to assign a degree to each node. When a node without degree is found, the random-number generator provides an $i$ and from the $N(i)$ curve a degree is assigned. It can be seen that this process prefers to assign lower degrees.

3. Link placement is done in two phases. The first phase connects all the nodes of degree one. Since the relative geographical position of each node is known, each node is connected to its nearest neighbors. This procedure is supported by the fact that most of these single-link nodes represent small users (with relatively small computational power) whose main goal is to enter the network at its nearest connecting point. The second phase places the remaining links. The placement is done according to the following algorithm: A node is picked at random. Then, another node is picked at random, and its distance $d$ is calculated. This distance is compared against a distance $d_1$ produced by an exponential function of the type

$$d_1 = \sqrt{\frac{3}{\pi}} e^{-x/k}, \quad x \geq 0,$$

where $x$ is generated by the random-number generator and the constant $k$ is supplied by the user. If $d > d_1$, the link is placed; otherwise another node will be selected. Note that once the node is connected by a number of links equal to its connectivity degree, it is removed from the pool of available nodes. Note also that the largest distance is $\sqrt{3}$, a diagonal of the 2 by 1 polygon that we chose to model the U.S.

After all links are placed, a depth-first search algorithm is run to ensure the network’s connectivity. If the network is found to be disconnected, the whole procedure is repeated. An example of a network generated by this program with 100 nodes is shown in Figure 2.

4. Finally, to each link a cost figure is assigned. The program is using Table I for its pricing policy. Generally, links connecting smaller nodes are assumed to be of lower baud rate, while large nodes are always connected by high-speed lines. All the networks are assumed to be medium loaded, giving equal selection chances to all three line-loading cases given in Table I.

Figure 2—Plots of synthesized 30, 50, and 100-node networks
NETWORK REPRESENTATION HEURISTICS

This section describes the class of techniques that compress the representation of the network topology, and that require no changes to the actual routing algorithm. A small amount of work is required before each routing in order to initialize the topology representation. These techniques are not heuristics, because the final routed paths are identical to those produced without topology compression.

Nodes of degree one or two usually contribute very little to the topology of the final path found by a routing algorithm. For example, no more than two nodes of degree one will ever be in the final path; similarly, nodes of degree two can be thought of as unnecessarily splitting a link into two pieces. A large portion of the nodes in a typical network are of degree one or two, suggesting that these nodes be removed from the main representation of the network and temporarily reinserted on an individual basis, as required, before routing. This approach can be applied to any type of network that possesses similar characteristics. The shortest-path problem is then redefined as: We consider a network of \( N \) nodes, some or all pairs of which are connected by links; the cost of each link is given. We restrict ourselves to the case where at least one path exists between any two nodes. Once each time the network topology changes, the network representation must be preprocessed into three disjoint sets of nodes according to their degree.

Preprocessing

1. **Isolated Node Removal**: Nodes originally of degree one are removed along with their connecting link. The information necessary to reattach them later, individually, is saved in a separate data structure.

2. **Chain Collapsing**: All nodes originally of degree two are removed; the two links associated with each are replaced by a single link whose cost is the sum of the original links. The information necessary to reattach them later, individually, is saved in a separate data structure.

Routing

3. **Topology Initialization**: Replaces only those removed nodes (and associated links) that correspond to terminal nodes for the desired path.

4. **Minimal-Cost Path**: Finds the path of minimum total cost between the two given terminal nodes by applying any of the classical routing algorithms.

RESULTS

A classical Dijkstra’s algorithm was used as a standard for measuring the performance improvement due to restructuring the network representation as described above. Software counters measured the number of references made by the algorithm to data structures that represent the topology and link costs of the network, first for the original network \( L \), and then for the reduced network \( L' \). The overhead needed to reconstruct the network, if any of the terminal nodes happened to be of degree one or two, was included in a count obtained during the routing. The results were averaged over 1,000 node pairs selected at random as the terminal nodes to be routed. Modeled networks with 15, 50, 100, 150, 200, 250, and 300 nodes and average connectivities of 2.5, 3, 4, and 5 were considered. Average routing time improvements, measured as the computation ratio \( M(L')/M(L) \) of memory references are plotted in Figure 3.

This preprocessing technique showed a clear improvement, saving on the average 15 to 35% of the memory references. The actual saving is very dependent on the particular network’s topology as seen in Figure 3. Notice that the improvement is best for the networks containing smaller numbers of nodes; such networks inherently have smaller values of average connectivity, implying a higher percentage of nodes of degree one or two.

ROUTING ALGORITHM HEURISTICS

This section describes the class of techniques that require major modifications of the routing algorithm. These techniques are truly heuristics in the sense that the final routed paths are considered *good* instead of *best*; that is, the paths may have a slightly higher cost than those found by the classical algorithms, but offer significant savings in routing time.

COST-RESTRICTED ROUTING

Inherent to minimal-cost routing is the fact that a majority of the paths go through the lower-cost links. A high cost indicates an overloaded link; thus the desirability of using it is very small. This suggests that the topology could be preprocessed each time the link costs are reassigned in order to remove the very high cost links from the database. The shortest-path problem is then defined as: We consider a network of \( N \) nodes, some or all pairs of which are connected by links; the cost of each link is given. We restrict ourselves to the case where at least one path exists between any two nodes. Each time the link costs are reassigned, the network representation must be preprocessed into two disjoint sets of links according to their associated costs.

Preprocessing

1. **Costly Link Removal**: Reduce the degree of those nodes whose degree is greater than a specified value \( D \) by removing a specified number \( d \) of their most costly links.

Routing

2. **Minimal-Cost Routing**: Try to find the path of least total cost between the two designated terminal nodes via the remaining links by applying any of the classical routing algorithms.
Figure 3—Computation ratio $M(L')/M(L)$ versus number of nodes $N$ for various values of average connectivity. The upper set of points in each plot correspond to isolated node removal only; the lower sets correspond to combined isolated node removal and chain collapsing. Average connectivity: (a) 2.5; (b) 3.0; (c) 4.0; (d) 5.0.
Figure 4—Cost restricted routing: Computation ratios $M(L')/M(L)$ and relative cost $C(L)/C(L')$ versus nodal degree for $N = 100$ (a), $200$ (b), and $300$ (c).
3. **Recovery Strategy**: If no path can be found, the routing algorithm is restarted using the original network representation. (Note: Other recovery strategies are of course possible, but our results do not justify a more complicated approach.)

**Results**

Long-haul networks of 100, 200, and 300 nodes were modeled based on TYMNET's cost assignment algorithm and nodal distribution; a typical average connectivity of 4.0 was used with a maximum allowed nodal degree of 16. A classical Dijkstra's algorithm was used as a standard against which we measured the amount of routing computation and the relative cost of the final path. Software counted the number of references made by the algorithm to the data structures in memory that represent the topology and link costs of the network, first for the original network $L$, and then for the reduced network $L'$. The software also kept track of the total path cost found during the routing. The results were averaged over 1,000 node pairs selected at random as the terminal nodes to be routed. Average routing-time improvements, measured as the computation ratio $M(L')/M(L)$, and the relative cost of the final path, $C(L)/C(L')$, are plotted in Figure 4.

Three comments about Figure 4 are appropriate: (1) Each of the plotted $M(L')/M(L)$ curves corresponds to a particular value of links removed, $d$. For each such curve, the value plotted at nodal degree $i$ corresponds to the removal of $d$ links from all nodes of degree $i$ or greater. (2) Since there were no nodes of degree 17 or greater, points at nodal degree of 17 represent the original unmodified network, $L$. The $M(L')/M(L)$ ratio is not 1.0; this is expected, since the links attached to each node were sorted before removal in the ascending order of their corresponding costs. The computation overhead necessary to perform the sorting operation is included in the total computation $M(L')$. As shown, this minor preprocessing alone is responsible for an 8% improvement. (3) The relative cost $C(L)/C(L')$ was nearly 1.0, until, in this removing procedure, the entire network consisted of nodes of degree five or less. This implied that the most important links attached to any node are the five cheapest.

For a particular network, a network designer needs to obtain a set of curves like the ones shown in Figure 4. From such set of curves, he can determine a set of values $d(n)$ of the number of links that can be removed from nodes of degree $n$ while keeping the final path cost near that of the "best" solution, and providing the best overall routing time. Such an approach was used on our experimental network, producing the results shown in Figure 5. It shows that a 25 to 35% improvement can be achieved at a cost penalty of less than 10%.

**DEGREE-RESTRICTED ROUTING**

Figure 6a shows that the innermost links of TYMNET routed paths tend to pass through nodes of relatively high degree. For example, about 80% of an average path consists of nodes of degree six or greater; even if the threshold is raised from six to nine, such nodes still represent 50% of the nodes visited by typical routed paths.

This result may have been due to the fact that nodes of high degree in TYMNET are usually major switching centers in which the attached links offer high-speed (low assigned link
The combination of these two routing steps is called degree-restricted routing rather than backbone routing because it is descriptive of the strategy, and so that the requirement of a network originally designed around a backbone is not implied. However, the adjective “backbone” will be used as a convenient means of identifying the extracted subnetwork consisting of nodes of high degree. The shortest-path problem is then defined as follows: We consider a network of $N$ nodes, some or all pairs of which are connected by links; the cost of each link is given. We restrict ourselves to the case where at least one path exists between any two nodes. Each time the topology changes, the network representation must be preprocessed into two disjoint sets of nodes according to their degree.

### Preprocessing

1. **Backbone Extraction**: A backbone subnetwork $B$ consisting of all the nodes of a specified degree $D$ or greater, together with those links which are connected to exactly two of these nodes, is extracted from the original network $L$.

### Routing

2. **Part I (Terminal Nodes to Backbone Nodes)**: For each of the terminal nodes, if it is a backbone node (degree of $D$ or greater), provide it as a starting node for Part II without further computation. Otherwise, find all paths of a specified limited depth using minimal-cost routing from the terminal node into the network $L$. In so doing, try to collect a set of nodes of degree $D$ or greater that are encountered along such paths, and provide this set of starting nodes to Part II. If no such nodes are found, go directly to step 4.

3. **Part II (Backbone Nodes to Backbone Nodes)**: The two sets of large nodes obtained in Part I are to be connected by a minimal-cost path within the backbone subnetwork $B$. If these two sets share one or more nodes in common, there already exists at least one path between the terminal nodes; select the minimal-cost path from this set of path(s). If the two sets of large nodes have no node in common, then routing between these sets is done using any of the classical routing algorithms modified to find the lowest-cost path between several alternative sources and destinations, restricting the search to the backbone subnetwork.

4. **Recovery Strategy**: If no nodes of degree $D$ or greater are found in Part I, the routing process is restarted using the original network $L$ and any of the classical algorithms.

(Note: Other recovery strategies are of course possible, but our results do not justify a more complicated approach.)

The maximum routing depth allowed in Part I must be sufficient to enter the backbone and collect a reasonable number of initial routing nodes for Part II. Figure 6 suggests that this depth need only be a couple of links; in our experiments, we used a conservative depth of three. It is actually desirable to have different values for $D$ in Part I and Part II. In Part II, a value of $D$ large enough to restrict the total size of the backbone subnetwork is needed; however, in Part I this value may be too large to reliably collect even a single entry to the backbone. Using different values of $D$ in the two parts implies

<table>
<thead>
<tr>
<th>Degree of a node</th>
<th>No. of nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
</tr>
</tbody>
</table>

The two

<table>
<thead>
<tr>
<th>Degree of a node</th>
<th>No. of nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
</tr>
</tbody>
</table>
Figure 6—Average nodal degree as a function of normalized path length for 1,000 random node pairs routed with (a) TYMNET cost assignment, and (b) random cost assignment.
that the Part II must accept initial starting nodes whose degree may be smaller than its own subset of nodes, with the agreement that the recovery procedure must be invoked if none of the initial nodes can be routed to a backbone node with a single link. Trying to explore this extra level in Part I would require search of a much larger topology, and thus represent much more computation.

The routing algorithm developed first checked the nodal degree of each terminal node. If both nodes, according to their nodal degree, belonged to the backbone part B, their routing was done strictly through the set of the backbone nodes. If one node belonged to the backbone B and the other was outside B, the outside node is first routed as described in Part I; collected nodes are then routed with the second terminal node as described in Part II.

Lastly, if both nodes were positioned outside the backbone, one node is routed as described in Part I. Two different cases can arise: (1) if in this routing process the other terminal node is encountered, Part I will continue until its normal termination and then the minimal path will be chosen. In this special case the backbone nodes are not used for routing. It was noted that a significant number of nodes were routed like this, which is a consequence of the fact that most of communications are local. (2) If the other terminal node is not encountered during the routing of the first node, Part I is repeated with the second terminal node. After its completion, Part II is used as already described.

Results

In order to determine suitable values for D in Part II, several trial values of D were selected and used to extract a corresponding backbone subnetwork B from a modeled network L of 300 nodes. For each trial value, 1,000 pairs of backbone nodes were randomly selected and routed within L and then B. The average computation ratio $M(B)/M(L)$ and relative cost $C(L)/C(B)$ for various values of D are shown in Figure 7.

In order to determine a suitable value for D in Part I, several trial values of D were selected and used as a criterion for collecting backbone entry nodes. For each trial value of D, another 1,000 pairs of nodes, one from B and another from $L-B$ were selected and routed. Average computation ratio $M(B)/M(L)$ and relative cost $C(L)/C(B)$ for various values of D are also shown in Figure 7.

The results confirm that different values of D should be used for the two parts of the algorithm. To minimize the errors, a value of 6 was chosen for Part II and 5 for Part I. After these values had been selected, another 1,000 random pairs of node pairs from L were routed using the degree-restricted routing algorithm defined above and compared to routing using the classical algorithm. Note that if the recovery strategy is used, the computation complexity is determined by summing the total number of references made by the failed routing process and the number of references made during the recovery procedure. The overall improvement in memory references was 48.52% with only a 2.32% relative increase in cost of the final path.

SUMMARY

This paper described three new approaches to the problem of finding low-cost paths in computer communication networks. Further information on the details of the actual experiments may be found elsewhere. The simulation results have demonstrated that algorithms based on these ideas produce excellent performance at nearly minimal cost for medium and large networks. Since the algorithms use only the network's topology and link costs (without necessarily knowing that the graph represents a computer network), similar results can be achieved for other networks that possess similar topological properties, and such heuristic solutions will be even more important when these same networks grow beyond 1,000 nodes.

The domain of the routing algorithm can be compressed by unconditionally removing nodes of degree one and two, producing a simpler network of fewer nodes and higher average connectivity. This technique provides a considerable improvement in speed with no increase in the cost of the final path; it is applicable to any type of network. Removing this small set of nodes (about a third of the total) resulted in a significant reduction of the computation required for the routing, since the required computation is proportional to the square of N.

Cost-restricted routing yields improvements dependent on the ratio of eliminated links to the total number of links, but it also may result in a small increase in the final path cost. The choice of a particular threshold or thresholds of cost for ignoring links will depend on the network, and should be determined from curves similar to those shown in Figure 4. Note that the results presented did not include a computation overhead for restructuring the database after the loading situation throughout the network is significantly changed. A separate experiment was run to include this overhead; 3% of the improvement is lost if the restructuring is done after every 20 routings. This drops to below 1% when the restructuring is done after every 50 routings. In practice, the loss will be somewhere in between, depending on the part of the day the network is used and on its loading.

Degree-restricted routing, like cost-restricted routing, requires preprocessing. However, experiments have shown that degree-restricted routing is more sensitive to connectivity than cost assignment is; thus the backbone extraction need only be performed when a link or node is actually removed or added. Moreover, the selection of value(s) for D reflects average characteristics of the network's topology, and is fairly insensitive to minor topology changes; thus the value(s) of D may be recalculated very infrequently (weeks for example).

As a final experiment, a combination of the node removal and degree-restricted routings was applied to the same 300-node network. The average saving in memory references was 54.72% with an average increase of 1.21% in the cost of the final path. This is an additional 6.20% improvement over the results obtained when only the degree-restricted routing algorithm was used. The distribution of number of performed routings versus achieved improvement is given in Figure 8. The values for D were the same as in the experiment pictured in Figure 7. When 1,000 node pairs were routed, in only 8
Figure 7—Degree restricted routing: Average computation ratio $M(B)/M(L)$ and relative cost $C(L)/C(B)$ versus nodal degree used as a breakpoint between large and small nodes. The points marked with an "x" represent large node to large node routing; those marked by a "+" represent large node to small node routing.
Figure 8—Distribution of NR (number of routings) versus achieved Saving (given in %), when degree restricted routing algorithm was applied (a) on unmodified 300-node network, and (b) on the same network but with nodes of degree one and two removed.

cases did the combined algorithm fail to find a path and have to use the help of the recovery strategy.

ACKNOWLEDGMENTS

The authors are indebted to Mr. J. Rindi and Mr. A. Rajaraman, both from TYMSHARE Corporation, for long and constructive discussions, and to the University of Santa Clara for their generous policy of free and unlimited computer time for students and faculty.

REFERENCES