1986

Space-Efficient Message Routing in c-Decomposable Networks

Greg N. Frederickson
Purdue University, gnf@cs.purdue.edu

Ravi Janardan

Report Number:
86-615

SPACE-EFFICIENT MESSAGE ROUTING IN c-DECOMPOSABLE NETWORKS

Greg N. Frederickson
Ravi Janardan

CSD-TR-615
July 1986
Revised December 1986

SPACE-EFFICIENT MESSAGE ROUTING IN c-DECOMPOSABLE NETWORKS

Greg N. Frederickson *
Ravi Janardan †

Department of Computer Science
Purdue University
West Lafayette, Indiana 47907

* The research of this author was supported in part by the National Science Foundation under grant DCR–8320124, and by the Office of Naval Research on contract N 00014-86-K-0689.

† The research of this author was supported in part by the National Science Foundation under grant DCR–8320124.

Abstract

Message routing strategies are given for any network that can be decomposed by a separator of size at most a small constant c into two networks with the same property. These strategies use a total of $O(cn \log n)$ items of routing information, keep node names to $O(\log n)$ bits, and still route along near-shortest paths. If the names generated from a separator-based hierarchical decomposition of the network alone are used, then the routing between any pair of nodes is along a path that is at most 3 times longer than an optimal path. By augmenting the node names with $O(c \log c \log n)$ additional bits, the length of any message path is improved to $(2/\alpha) + 1$ times the length of an optimal path, where $\alpha > 1$ is the positive root of the equation $\alpha^{\lceil (c+1)/2 \rceil} - \alpha - 2 = 0$.

Keywords and phrases. Distributed network, graph theory, k-outerplanar graph, routing, separator, series-parallel graph, shortest paths.

1. Introduction

One of the primary functions in a distributed network is the routing of messages between pairs of nodes. Assuming that a cost, or distance, is associated with each edge, it is desirable to route along shortest paths. While this can be accomplished using a complete routing table at each of the n nodes in the network, such tables are expensive for large networks, storing a total of $\Theta(n^2)$ items of routing information. Recent research has focused on identifying classes of network topologies for which the shortest paths information at each node can be stored succinctly, if suitable names are assigned to the nodes. Optimal routing schemes using a total of $\Theta(n)$ items of routing information have been given for networks such as trees, unit-cost rings [SK,vLT1], complete networks, unit-cost grids [vLT2], and networks at the lower end of a hierarchy identified in [FJ1]. Unfortunately, the approach in the above research does not yield a space-efficient scheme for even such a simple class as the series-parallel networks. However, by shifting our focus to consider schemes that route along near-shortest paths, we have been able to generate space-efficient strategies for much broader classes of network topologies.

In this paper we present a near-optimal routing scheme that handles any network that can be decomposed by a separator of size at most a constant c into two networks with the same property. Examples of such networks are the series-parallel networks, for which $c = 2$, and the k-outerplanar networks, for $k > 1$ a constant, for which $c = 2k$. We use a separator strategy to hierarchically decompose the network and generate names for the nodes. Our solution uses $O(\log n)$-bit names and $O(cn \log n)$ items of routing information overall, where each item is $O(\log n)$ bits long. If the names derived from the decomposition alone are used, then the length

of the routing between any pair of nodes is at most 3 times that of a corresponding optimal routing. By augmenting the node names with $O(c \log c \log n)$ additional bits, we reduce this bound to 2 for $c \leq 3$ and to a value ranging from 2 to 3 as c increases.

In related work [FJ2], two efficient schemes are given for general planar networks. The first scheme uses $O(\log n)$-bit names and $O(n^{4/3})$ items to achieve a bound of 3 on the routings. For any constant ϵ, $0 < \epsilon < 1/3$, the second scheme can be set up to use $O(n^{1+\epsilon})$ items and achieve a bound of 7 on the routings, but at the expense of $O((1/\epsilon) \log n)$-bit names. These approaches also rely on separators [LT,M] to decompose the network and generate the node names. However, owing to the comparatively larger size of the separator, the techniques used are somewhat different from those in the current paper.

A preliminary version of this paper appeared in [FJ3].

2. Preliminaries

Our approach can handle any class of c-decomposable graphs, defined as follows. Let G be a graph with nonnegative weights on its nodes and let c be a constant. A *c-separator* for G is a set C of at most c *separator nodes* whose removal partitions the remaining nodes into sets A and B, each containing at most two-thirds the total weight and with no node in A adjacent to a node in B. If G has a c-separator for every assignment of weights to its nodes, then so does every proper subgraph of G. We say that G is *c-decomposable.* Examples of classes of c-decomposable graphs are the series-parallel graphs ($c = 2$) and the k-outerplanar graphs, for $k > 1$ a constant ($c = 2k$). We give linear-time separator strategies for these classes in a later section.

3. Hierarchical decomposition and naming

Given a c-decomposable graph G we first show how to hierarchically decompose it and generate the node names. The graph at level 0 is $G_0 = G$. Equal positive weights are assigned to its nodes and it is then separated into the subgraphs G'_{00} and G'_{01} induced on $A \bigcup C$ and $B \bigcup C$ respectively. To preserve distances between nodes in C, each of these graphs is then augmented by a graph of size $O(c^4)$ derived from the other graph. This yields the level 1 graphs G_{00} and G_{01}. Let G_ω be a level $i > 1$ graph, where ω is a binary string, and let the *core* of G_ω be the subgraph of G_ω excluding the nodes and edges introduced by augmentations done so far. If the core of G_ω has more than c nodes, then a pair of level i graphs $G_{\omega 0}$ and $G_{\omega 1}$ is generated from G_ω as before, except that nodes not in the core of G_ω are given weight zero when finding a separator. Clearly, there are $O(\log n)$ levels in the decomposition.

In general, let G_ω be a level i graph for any $i > 0$. If G_ω is decomposed further, then any separator node v of G_ω belonging to its core is a *level i node*. Otherwise, the core of G_ω contains at most c nodes and each core node is a *level i node*. For ω' a proper prefix of ω and of length j, each separator node u of $G_{\omega'}$ is an *ancestor of v for level j*. If u is in the core of $G_{\omega'}$, then it is a *core ancestor* of v. We call v a *descendant* of u. Two level i nodes in the core of the same level i graph are *siblings*. Two nodes of G are *related* in the decomposition if one is a core ancestor of the other or if they are siblings.

Each level i node of G_ω is given the name ω, along with an integer distinguisher that is at most c. Clearly, the names are $O(\log n)$ bits long. This naming has the property that two nodes are related if and only if the distinguisher-free portions of

their names are identical or if one is a proper prefix of the other. For unrelated nodes v and u, if l is the length of the longest common prefix, then v and u are in the core of the same level $l-1$ graph, but are in the cores of different level l graphs. Level l is called the *separating level for v and u.* It is used to determine an appropriate core ancestor of the two nodes through which messages can be routed.

We now discuss how the augmentation is performed. Let $G'_{\omega 0}$ and $G'_{\omega 1}$ be the graphs resulting from the separation of G_ω. $G'_{\omega 0}$ is augmented as follows to obtain $G_{\omega 0}$. Let C be the set of separator nodes of G_ω. A graph that is the union of the shortest path trees T_v in G_ω from each node v in C to the nodes in $C-\{v\}$ is constructed. The induced subgraph of this graph restricted to $G'_{\omega 1}$ is inferred, and then contracted by repeatedly replacing each degree 2 node not in C and its incident edges by an edge of cost equal to the sum of the costs of the two edges removed. Graph $G_{\omega 0}$ is the union of the contracted graph and $G'_{\omega 0}$. $G_{\omega 1}$ is similarly obtained from $G'_{\omega 1}$. In the augmented graphs, the distance between any two core nodes is equal to the distance between them in G. Furthermore, for the chosen assignment of zero weights to the noncore nodes and equal weights to the core nodes, both $G_{\omega 0}$ and $G_{\omega 1}$ have c-separators.

As the following lemma shows, $G_{\omega 0}$ and $G_{\omega 1}$ are not too large compared to $G'_{\omega 0}$ and $G'_{\omega 1}$, respectively.

Lemma 1. The augmentation introduces fewer than c^4 nodes and $3c^4/2$ edges into each of $G'_{\omega 0}$ and $G'_{\omega 1}$.

Proof. We prove the claim for $G'_{\omega 0}$. Let J_0 be the graph with which $G'_{\omega 0}$ is augmented to obtain $G_{\omega 0}$. In worst case J_0 is the contraction of the union of c shortest path trees T_v. There are at most $c(c-1)$ shortest paths in these trees, and

for each path there is a corresponding contracted shortest path in J_0. If two shortest paths in J_0 meet, then they share a maximal subpath. We call the endpoints of this subpath *meeting nodes*. In worst case there are $c(c-1)(c(c-1)-1)/2 < c^2(c-1)^2/2$ meetings between different pairs of shortest paths.

We derive an upper bound on the sum of the degrees of the nodes in J_0. Starting with an empty graph, union in the shortest paths of J_0 one at a time. Assign each node a degree when it is introduced into the graph for the first time. Assign it degree 1 if it is in C, and degree 2 otherwise. Taken over all nodes in J_0, this contributes $2(\mid V(J_0) \mid -c) + c = 2 \mid V(J_0) \mid -c$ to the degree sum. If two shortest paths meet, then increase the degree of each of their meeting nodes by 1. Thus the increase in the degree sum due to all meetings between shortest paths is less than $c^2(c-1)^2$. Thus the degree sum is less than $2 \mid V(J_0) \mid -c + c^2(c-1)^2$.

Now each node in $V(J_0) - C$ has degree at least 3, so that the degree sum is at least $3(\mid V(J_0) \mid -c) + c = 3 \mid V(J_0) \mid -2c$. It follows that $\mid V(J_0) \mid -c < c^2(c-1)^2 < c^4$.

The number of edges in J_0 is half the degree sum of J_0. Thus there are less than $\mid V(J_0) \mid -c/2 + c^2(c-1)^2/2 < 3c^4/2$ edges in J_0. ∎

Since a node introduced by the augmentation can later become a separator node, we must distinguish between two types of ancestors, core and noncore. If u is an ancestor of v, but not a core ancestor, then we call u a *noncore ancestor of* v.

For the purposes of routing, we associate with each ancestor u of v a core ancestor of v, called the surrogate for u at v and denoted $surrogate_v(u)$. If u is a core ancestor, then we take $surrogate_v(u)$ to be u itself. If u is a noncore ancestor, then let j be the separating level for v and u, and let u' be a common ancestor for

level j on a shortest (v,u)-path in the level $j-1$ graph whose core contains both v and u. Then $surrogate_v(u)$ is just $surrogate_v(u')$.

4. Routing information at the nodes

A routing table is maintained, giving for each related node u the name $next\_node_v(u)$ of the next node on a shortest (v,u)-path in G. To route to u, v sends the message to $w = next\_node_v(u)$ over edge $\{v,w\}$. If w and u are unrelated, then w will not have shortest paths information for u. So v makes available to w in the message header, the name $milestone_v(u)$ of a common core ancestor of w and u through which the routing can proceed. Let j be the separating level for w and u, and let y be a common ancestor for level j on a shortest (w,u)-path in the level $j-1$ graph whose core contains both w and u. Then $milestone_v(u)$ is just $surrogate_w(y)$. The name $milestone_v(u)$ is stored at v.

Information is also stored at v to enable routing to unrelated nodes. Let v be a level i node. For $j < i$, if a is the closest ancestor of v for level j, then $surrogate_v(a)$ is stored at v for level j. If v is the source of a message whose destination is an unrelated node u, then v routes to u via the $surrogate_v(a)$ stored for the separating level of v and u.

The following theorem bounds the amount of routing information in the network.

Theorem 1. For any n-node c-decomposable graph, the above scheme uses a total of $O(cn \log n)$ items of routing information.

Proof. Since each node has $O(c \log n)$ core ancestors, $O(cn \log n)$ items are stored for them overall. Since each node is a descendant of $O(c \log n)$ core ancestors,

$O(cn \log n)$ items are maintained overall for descendants. Since each node has at most c siblings, $O(cn)$ items are maintained overall for siblings. ∎

5. The routing strategy

The routing from a source s to a destination d is as follows. The message header contains separate fields for the milestone and the destination, both initially set to d. The milestone field alone is reset, as necessary, during the routing. Let d' denote the current name in the milestone field. Each node v participating in the routing performs a *routing action*. This involves determining $w = next\_node_v(d')$, resetting d' to $milestone_v(d')$ if w and d' are unrelated, and then sending the message to w.

If s and $d' = d$ are unrelated, then s performs a routing action. Otherwise, let l be the separating level for s and d, and a the closest ancestor of s for level l. Node s resets d' to $surrogate_s(a)$ and performs a routing action. The two cases can be distinguished using the names s and d. Each intermediate node different from the current d' will find the latter in its routing table and thus can perform a routing action. Eventually the message reaches the current d'. If d' is d, then the routing terminates. Otherwise d' is reset to d and a routing action is performed. Note that since the milestone field is always reset to a core ancestor of d, the previous milestone need not be saved.

The following theorem provides an upper bound on the length of the routings achieved.

Theorem 2. Let G be a c-decomposable graph. For any nodes s and d, let $\rho(s,d)$ denote the length of a shortest (s,d)-path in G and let $\hat{\rho}(s,d)$ denote the length of the (s,d)-path generated in the above scheme. Then the *performance bound* of the

scheme is $\hat{\rho}(s,d)/\rho(s,d) \leq 3$.

Proof. If s and d are related then the routing is along a shortest (s,d)-path. This is because every node participating in the routing performs a routing action with respect to the milestone, which is always on a shortest (s,d)-path. Otherwise, let a' be $surrogate_s(a)$, and consider the first occasion a milestone a'' is reached, where a'' is possibly a'. Since a'' is a common core ancestor of s and d, by the above reasoning the routings from s to a'' and from a'' to d are both along shortest paths. Thus

$$\begin{aligned}
\hat{\rho}(s,d) &= \rho(s,a'') + \rho(a'',d) \\
&\leq \rho(s,a'') + \rho(a'',a') + \rho(a',d) \\
&= \rho(s,a') + \rho(a',d), \text{ since } a'' \text{ is on a shortest } (s,a')\text{-path} \\
&\leq \rho(s,a') + \rho(a',a) + \rho(a,d) \\
&= \rho(s,a) + \rho(a,d), \text{ since } a' \text{ is on a shortest } (s,a)\text{-path} \\
&\leq \rho(s,a) + \rho(a,s) + \rho(s,d) \\
&\leq 3\rho(s,d), \text{ since, by our choice of } a,\ \rho(s,a) \leq \rho(s,d). \ \blacksquare
\end{aligned}$$

In fact, the bound of 3 is approachable. Let a^* different from a be the ancestor on a shortest (s,d)-path and let $a'' = a' = a$. Let $\rho(s,a) = \rho(s,d) - \rho(a^*,d)$, and let $\rho(a,d) = \rho(a,s) + \rho(s,d) = 2\rho(s,d) - \rho(a^*,d)$. Then $\hat{\rho}(s,d)/\rho(s,d) = (3\rho(s,d) - 2\rho(a^*,d))/\rho(s,d)$ approaches 3 as $\rho(a^*,d)$ becomes vanishingly small.

6. Improving the performance bound

In the previous scheme, a performance bound of 3 was obtained by choosing the closest ancestor of s. To improve this bound, we encode additional information

in the node names and use it to refine the choice of an ancestor. The rest of the routing strategy is the same.

Let v be a level i node, $i \geq 1$. For each $j < i$, instead of storing at v the name $surrogate_v(a)$ for the closest ancestor of v for level j, we store the names u and $surrogate_v(u)$ for each ancestor u of v for level j. This introduces a total of $O(cn \log n)$ additional items.

Node v's name is augmented with information about the relative magnitudes of its distances from its ancestors for level j. Two pieces of information are encoded for each ancestor, with the information for different ancestors appearing in the lexicographic order of the names assigned to them from the decomposition. The first specifies its position in an ordering of the ancestors by nondecreasing distances from v, with ties broken lexicographically. The second piece of information is as follows. Let $\alpha > 1$ be a function of c to be specified later. For each ancestor a' with index p' in the above ordering by distances, let a'' be the ancestor with the smallest index $p'' > p'$, such that $\rho(v, a') \leq (1/\alpha)\rho(v, a'')$. Then, in addition, p'' is encoded in v's name for a'. If a'' does not exist, then zero is recorded for a'. All this information can be encoded using at most $2c \log c$ bits per level j. As there are at most $\log_{3/2} n = 1.71 \log n$ levels, the total number of additional bits encoded into v's name is $3.42 \log c \log n$.

From the ancestors of s and d for separating level l, an appropriate ancestor is chosen at s as follows. Clearly, if there are ancestors a' and a'' such that $\rho(s, a') < \rho(s, a'')$ and $\rho(d, a') < \rho(d, a'')$, then a'' can be eliminated. Using the information encoded in its name and that of d, s determines a subset of the ancestors in which no ancestor eliminates another. (These ancestors will be known in terms of

their positions in the above lexicographic ordering. However, s can determine their names, since the names of its ancestors at each level are available.) Let $a_1, a_2, \ldots, a_h$ be the $h \leq c$ such ancestors, indexed in increasing order of their distances from s. Denote $\rho(s, a_i)$ by x_i and $\rho(d, a_i)$ by y_i, $1 \leq i \leq h$. Thus $x_1 < x_2 < \cdots < x_h$. Furthermore, since no ancestor eliminates another, we have $y_1 > y_2 > \cdots > y_h$.

Let m be an integer parameter, $1 \leq m \leq h$, to be specified later. If there exists a minimum index i, $1 \leq i < m$, such that $x_i \leq (1/\alpha)x_{i+1}$, then s chooses a_i. Otherwise, if there exists a maximum index i, $m < i \leq h$, such that $y_i \leq (1/\alpha)y_{i-1}$, then s chooses a_i. Otherwise s chooses a_m. As demonstrated in the proof of the following theorem, the appropriate choice for m is $\lfloor (h+1)/2 \rfloor$.

Theorem 3. For any c-decomposable graph, the performance bound of the above scheme satisfies $\hat{\rho}(s,d)/\rho(s,d) \leq (2/\alpha) + 1$, where $\alpha > 1$ is the positive root of the equation $\alpha^{\lceil (c+1)/2 \rceil} - \alpha - 2 = 0$.

Proof. From the proof of Theorem 2, the length of the generated routing is at most the sum of the distances from s and d to the chosen ancestor. It follows that if there is a shortest (s,d)-path through this ancestor then the routing is optimal. Thus assume that there is no shortest (s,d)-path through the chosen ancestor, and that there is one through a_q, $1 \leq a_q \leq h$.

Case 1: a_i, $1 \leq i < m$, is chosen in the scan over the x's.

(a) Suppose that $i < q$. Then since $x_{i+1} \leq x_q$ and $x_i \leq (1/\alpha)x_{i+1}$, we have

$x_i \leq (1/\alpha)x_q$. Thus,

$$\begin{aligned}\hat{\rho}(s,d)/\rho(s,d) &\leq (x_i + y_i)/(x_q + y_q) \\ &\leq (2x_i + x_q + y_q)/(x_q + y_q), \text{ since } y_i \leq x_i + x_q + y_q \\ &\leq (2x_q/\alpha)/(x_q + y_q) + 1 \\ &\leq (2/\alpha) + 1.\end{aligned}$$

(b) Suppose that $i > q$. Since $x_j > (1/\alpha)x_{j+1}, 1 \leq j < i$, it can be shown inductively that $x_i < \alpha^{i-q}x_q$.

$$\begin{aligned}\hat{\rho}(s,d)/\rho(s,d) &\leq (x_i + y_i)/(x_q + y_q) \\ &< (\alpha^{i-q}x_q + y_q)/(x_q + y_q), \text{ from above, and since } y_q > y_i \\ &\leq \alpha^{i-q} \\ &\leq \alpha^{m-2}.\end{aligned}$$

Case 2: a_i, $m < i \leq h$, is chosen in the scan over the y's.

(a) If $i > q$, then similar to Case 1(a) it can be shown that $\hat{\rho}(s,d)/\rho(s,d) \leq (2/\alpha) + 1$.

(b) Suppose that $i < q$. Then $y_i < \alpha^{q-i}y_q$ holds. Similar to Case 1(b) it can be shown that $\hat{\rho}(s,d)/\rho(s,d) < \alpha^{h-m-1}$.

Case 3: a_m is chosen by default.

(a) If $m > q$, then we have $x_m < \alpha^{m-q}x_q$, and as in Case 1(b) it can be shown that $\hat{\rho}(s,d)/\rho(s,d) < \alpha^{m-1}$.

(b) If $m < q$, then we have $y_m < \alpha^{q-m}y_q$, and as in Case 1(b) it can be shown that $\hat{\rho}(s,d)/\rho(s,d) < \alpha^{h-m}$.

From the above it follows that

$$\hat{\rho}(s,d)/\rho(s,d) \leq \max\{(2/\alpha) + 1, \alpha^{m-1}, \alpha^{h-m}\}.$$

For $m = \lfloor (h+1)/2 \rfloor$ we have $\alpha^{m-1} \leq \alpha^{h-m} \leq \alpha^{\lceil (c-1)/2 \rceil}$. The larger of $(2/\alpha)+1$ and $\alpha^{\lceil (c-1)/2 \rceil}$ is minimized when $\alpha > 1$ is the positive root of $(2/\alpha)+1 = \alpha^{\lceil (c-1)/2 \rceil}$, i.e., $\alpha^{\lceil (c+1)/2 \rceil} - \alpha - 2 = 0$. ∎

For small values of c the above theorem yields performance bounds that are appreciably better than 3. For instance, if c is 2 or 3, then $\hat{\rho}(s,d)/\rho(s,d) \leq 2$; if c is 4 or 5, then $\hat{\rho}(s,d)/\rho(s,d) \leq 2.32$. These performance bounds are approachable. Let a_1 be the chosen ancestor and suppose that there is a shortest (s,d)-path through a_2. Let $x_1 = (1/\alpha)x_2$ and $y_1 = x_1 + x_2 + y_2$. Then $\hat{\rho}(s,d)/\rho(s,d)$ approaches $(2/\alpha)+1$ as y_2 becomes vanishingly small.

Figure 1 illustrates schematically the improved routing algorithm for a 5-decomposable graph. There are just four ancestors a_1, a_2, a_3 and a_4 to choose from, since the unnamed ancestor is eliminated by a_3. For this example, $\alpha \approx 1.52$, $m = 2$, and the shortest (s,d)-path is through a_2. In the routing algorithm, the scan over the x's is inconclusive. The scan over the y's first succeeds at y_3, since $y_3 = 6$ and $(1/\alpha)y_2 = 6.6$. Thus a_3 is chosen, yielding a routing that is $21/14 = 1.5$ times longer than the optimal.

7. Separator strategies

We give linear-time algorithms to find a 2-separator in series-parallel graphs and a $2k$-separator in k-outerplanar graphs. Assume without loss of generality that the graph is simple, i.e., any two nodes are joined by at most one edge, and biconnected. These conditions can be always be enforced, without affecting shortest paths, by deleting all but the least cost edge joining any pair of nodes, and by suitably introducing edges of large cost.

Series-parallel graphs

Two edges in a graph are *series* if they are the only edges incident with a node, and *parallel* if they join the same pair of nodes. A *series-parallel* graph is recursively defined as follows [D]. An edge is a series-parallel graph. The graph obtained by replacing any edge in a series-parallel graph either by two series edges or by two parallel edges is series-parallel. A *two-terminal series-parallel graph* is a graph with two distinguished nodes called *terminals* and is defined recursively as follows. Any edge is a two-terminal series-parallel graph, the terminals being its endpoints. If H_1 and H_2 are two-terminal series-parallel graphs, then so is the graph H obtained either by identifying one of the terminals of H_1 with one of the terminals of H_2 or by identifying them in pairs. In the former case the terminals of H are the unidentified terminals of H_1 and H_2, while in the latter they are the identified terminals. With every two-terminal series-parallel graph G can be associated a *binary structure tree* [VTL]. Each leaf of the tree represents an edge of G. If v is an internal node of the tree with children v_1 and v_2 representing the two-terminal series-parallel graphs H_1 and H_2, then v represents the two-terminal series-parallel graph H obtained as above from H_1 and H_2. The root of the tree represents G. Since every series-parallel graph is two-terminal series-parallel for an appropriate choice of terminals [D], a structure tree can be associated with it.

Given any assignment of nonnegative weights to the nodes of a series-parallel graph G, where the weights sum to unity, a 2-separator can be found as follows. Construct a structure tree for G with root r, as described in [VTL]. For each node x in the tree, let $W(x)$ be the sum of the weights of the nodes in the series-parallel graph represented by x. For each non-leaf node, let the *heavy child* be the one with the larger $W(\cdot)$, ties broken arbitrarily.

Initially, set x to r. While x is not a leaf of the structure tree and $W(x) > 2/3$, reset x to its heavy child. When this step terminates, let C be the set of terminals of the series-parallel graph represented by x. Let A consist of the remaining nodes of this graph and let B be $V(G) - (A \bigcup C)$. It may be verified that A, B and C satisfy the conditions for a 2-separator.

Theorem 4. A 2-separator of an n-node series-parallel graph can be found in $O(n)$ time.

Proof. The structure tree can be constructed and searched in $O(n)$ time. ∎

k-Outerplanar graphs

The k-outerplanar graphs are defined as follows [B]. Consider a plane embedding of a planar graph. The nodes on the exterior face are *layer* 1 nodes. For $i > 1$, the *layer* i nodes are those that lie on the exterior face of the embedding resulting from the deletion of all layer j nodes, $j < i$. A plane embedding is *k-outerplane* if it contains no node with layer number exceeding k. A planar graph is *k-outerplanar* if it has a k-outerplane embedding.

Let G be a k-outerplanar graph and G^* a k-outerplane embedding of G. Given any assignment of nonnegative weights to the nodes of G, where the weights sum to unity, a $2k$-separator can be found as follows. The interior faces of G^* are first triangulated. Each interior face whose boundary consists of nodes all with the same layer number is triangulated arbitrarily. Each interior face whose boundary consists of both layer i and layer $i+1$ nodes, $1 \leq i < k$, is triangulated by repeatedly adding an edge joining a layer $i+1$ node to a layer i node. The resulting embedding, G^*_Δ, is also k-outerplane, with each layer $i+1$ node adjacent to at least one layer i node. The desired separator is found in G^*_Δ.

At all times, the algorithm maintains a path P of length at most $2k$ in G^*_Δ, which disconnects G^*_Δ into two regions. The algorithm repeatedly modifies P until the total weight of the nodes in each region is at most 2/3. Initially, P consists of a single edge joining a pair of level 1 nodes. In general, P has layer 1 nodes as endpoints and is such that from one end to the other, the layer numbers of its nodes first increase monotonically and then decrease monotonically, possibly with a single pair of consecutive nodes of the same layer number.

For each region bounded by P, determine the sum of the weights of the nodes contained in the region. Let the *heavy region* be the one with the larger total weight, ties broken arbitrarily. If the heavy region has weight exceeding 2/3 then modify P as follows.

Let v be a node on P of highest layer number and u the neighbor of v on P with the higher layer number, ties broken arbitrarily. Let P_1 and P_2 be the subpaths of P on either side of edge $\{v, u\}$, where v is an endpoint of P_1 and u an endpoint of P_2. Consider the face in the heavy region whose boundary contains edge $\{v, u\}$ and let w be the third node on this face. For some i, $1 \leq i < k$, the layer numbers of v, u and w must each be either i or $i+1$. There are two cases of interest.

If the layer number of w exceeds the layer number of at least one of v and u, then reset P to the path consisting of P_1, $\{v, w\}$, $\{w, u\}$ and P_2. If the heavy region now has total weight exceeding 2/3 then modify P recursively.

Otherwise, let P_3 be a path in the heavy region from w to the exterior face such that the layer numbers of its nodes decrease monotonically. Such a path can be found because each layer $i+1$ node is adjacent to at least one layer i node, $1 \leq i < k$. Furthermore, P_3 can always be picked so that it is either node-disjoint

from both P_1 and P_2, or meets one of these paths at a node, and contains the segment of this path from the meeting point to the exterior face. Determine the total weight of the nodes contained in the region bounded by P_1, $\{v,w\}$ and P_3. Likewise for the region bounded by P_2, $\{u,w\}$ and P_3. Without loss of generality assume that the former region is the heavy region. If P_3 and P_1 share no nodes, then reset P to the path consisting of P_1, $\{v,w\}$ and P_3. Otherwise, let z be the first node common to P_1 and P_3 and let e be an edge incident on z from the cycle consisting of the (z,v)-subpath of P_1, the edge $\{v,w\}$ and the (w,z)-subpath of P_3. Reset P to the path consisting of P_1, $\{v,w\}$ and P_3, with e deleted. If the heavy region now has total weight exceeding $2/3$, then modify P recursively.

Eventually a path P is found such that the heavy region has total weight at most $2/3$. It can be shown inductively that P is a disconnecting path for G^*_Δ, hence for G^*, and has at most $2k$ nodes. Let C be the set of nodes on P, A be the set of the nodes in the heavy region and B be $V(G) - (A \bigcup C)$.

Theorem 5. A $2k$-separator of an n-node k-outerplanar graph can be found in $O(n)$ time.

Proof. Given the embedding G^*, represented using the data structure of [LT], the layer numbers can be computed in $O(n)$ time [B]. The triangulation can also be done in $O(n)$ time. The time to successively modify paths is as follows. Consider any path P in the algorithm. The node v of highest layer number is identified at the time P is formed. The nodes u and w can be identified in constant time.

If the layer number of w exceeds the layer number of at least one of v and u, then P can then be modified and the weight of the heavy region determined in constant time. The node of highest layer number on the resulting path is w. Charge

this cost to edge $\{v, w\}$, which is eliminated from the heavy region. Thus the total time for all paths modified in this fashion is $O(n)$.

Otherwise we find P and determine as follows which of the two regions, one bounded by P_1, $\{v, w\}$ and P_3, and the other by P_2, $\{u, w\}$ and P_3, is the heavy region. Accumulate the weight of the two regions by alternately examining one node from each region, stopping when one of the regions has been exhausted. To do this efficiently perform a depth-first search in each region in incremental fashion, i.e., search in one region until a node has been added to the depth-first search tree, then suspend the search in this region and resume it in the other region. As the graph is planar, the time for this is proportional to the size of the exhausted region. Since the weight of the exhausted region is known, the weight of the other region can be computed, and the heavy region determined. P is then reset appropriately. The node of highest layer number on the resulting path is one of v, u and w.

The time to thus modify P is proportional to the size of the exhausted region. Charge this cost to the nodes in the region that is not the heavy region. This results in constant charge per node. Since each of these nodes is charged at most once and then eliminated, the total time for all paths modified this way is $O(n)$. ∎

8. Set-up time for routing schemes

A class of graphs is *uniformly sparse and contractable* if for any graph in the class, every subgraph of the graph has a number of edges proportional to the number of nodes, and any contraction of the subgraph is also in the class. Consider any uniformly sparse and contractable class of c-decomposable graphs with a linear-time c-separator algorithm. For any n-node graph G from this class, we show that our schemes can be set up in $O(cn(\log n)^2 \div c^4 n \log n)$ time. If G is planar, then a setup

time of $O(cn(\log n)^{3/2} + c^2 n \log n + c^4 n(\log n)^{1/2})$ can be realized. Examples of such graphs G are the series-parallel and k-outerplanar graphs.

We first establish upper bounds on the total number of noncore nodes and core nodes generated in decomposing G down to graphs of core size at most c. Let $NC(n)$ be the total number of noncore nodes, counting each occurrence of a node as a noncore node. Thus

$NC(n) = 0$, for $n = c$

$NC(n) < NC(an) + NC((1-a)n) + c^4$, for $n > c$, where $1/3 \leq a \leq 2/3$.

By an inductive proof, $NC(n) \leq c^3 n - c^4$.

Let $C(n)$ be the total number of core nodes contained in all cores of size greater than c, counting each occurrence of a node as a core node. Thus

$C(n) = 0$, for $n = c$

$C(n) \leq C(an - c) + C((1-a)n - c) + n$

$< n + C(an) + C((1-a)n)$, for $n > c$, where $1/3 \leq a \leq 2/3$.

By an inductive argument, $C(n) \leq 1.09(n \log n - (\log c)n)$.

We first analyze the setup time for the scheme with performance bound 3. The time for doing the decomposition and naming is as follows. Let G_ω be any graph in the decomposition with a total of n_ω nodes. Note that G_ω is in the same class as G. If G_ω is a level $i-1$ graph with more than c core nodes, then the time to separate it and name the resulting level i graphs and level i nodes is $O(n_\omega)$. The augmentation of the level i graphs involves computing at most c shortest path trees in G_ω, which can be done in $O(cn_\omega \log n_\omega)$, i.e., $O(cn_\omega \log n)$ time, using the algorithm from [J]. As $n_\omega = core\_size(G_\omega) + noncore\_size(G_\omega)$, where $core\_size(G_\omega)$ and $noncore\_size(G_\omega)$ respectively denote the number of core and noncore nodes in

G_ω, the time for G_ω is $O(c \log n(core\_size(G_\omega) + noncore\_size(G_\omega))$. The time for all graphs G_ω is obtained by summing each term over the G_ω. From the bounds on $C(n)$ and $NC(n)$, this is $O(cn(\log n)^2 + c^4 n \log n)$. Any graph G_ω with c or fewer core nodes is not decomposed further. The total time to name the core nodes in all such graphs G_ω is $O(n)$.

The time for setting up shortest paths information between all related nodes is as follows. Let v be a level i node resulting from the decomposition of level $i-1$ graph G_ω. For each descendant (resp. sibling) u in the core of G_ω, $next\_node_v(u)$ can be determined by constructing a shortest path tree in G_ω, rooted at v. Simultaneously, $next\_node_u(v)$ can be set up for the core ancestor (resp. sibling) v. Arguing as before, the time for this is $O(cn(\log n)^2 + c^4 n \log n)$.

For a graph G_ω with c or fewer core nodes, $next\_node_v(u)$ can be set up for any pair v and u of core nodes in $O(cn_\omega \log n_\omega)$, i.e., $O(cn_\omega \log n)$. The time for all G_ω is $O(\sum_{\text{all } G_\omega} cn_\omega \log n)$, i.e., $O(c \log n \sum_{\text{all } G_\omega}(core\_size(G_\omega) + noncore\_size(G_\omega))$. The first term contributes $O(n)$ and the second $O(c^3 n)$. Thus the total time is $O(c^4 n \log n)$.

For any level i node v, the closest ancestor and the corresponding surrogate for each level $j < i$ can be found as follows. Consider the ancestors of v for level j. From the shortest path computations, the distance from v to each core ancestor is known. To compute the distance to a noncore ancestor a, let $j' < j$ be the separating level for v and a. Minimize $\rho(v, a') + \rho(a', a)$ over all common ancestors a' of v and a for level j'. Since $j' < j$, the distances $\rho(v, a')$ and $\rho(a', a)$ will be available for all ancestors a', whether core or noncore.

This process also yields $surrogate_v(a)$ for each noncore ancestor a. Let a' be

the ancestor for level j' that minimizes $\rho(v,a') + \rho(a',a)$. Then $surrogate_v(a)$ is just $surrogate_v(a')$. Recall that for a core ancestor, the surrogate is the ancestor itself. After the distances to all ancestors of v for level j have been determined, the names of the closest ancestor and the corresponding surrogate are stored at v.

The time to compute the distance from v to each noncore ancestor a for level j is $O(c)$, hence $O(c^2)$ for all noncore ancestors. Thus the time for all closest ancestor and surrogate computations at v is $O(c^2 \log n)$, hence $O(c^2 n \log n)$ at all nodes.

The milestone information can be set up as follows. Let v be a level i node resulting from the decomposition of a level $i-1$ graph G_ω. Let u be a core node of G_ω related to v, and suppose that $w = next\_node_v(u)$ and u are unrelated. Let j be the separating level for w and u, and y their common ancestor for level j minimizing $\rho(w,y) + \rho(y,u)$. Then $milestone_v(u)$ is just $surrogate_w(y)$. If $next\_node_u(v)$ and v are unrelated, then $milestone_u(v)$ can simultaneously be set up at u.

The time to set up $milestone_v(u)$ is $O(c)$ for each node u, hence $O(c\ core\_size(G_\omega))$ for all u in the core of G_ω. Since there are $O(c)$ level i nodes v for G_ω, the milestones at all these nodes can be set up in $O(c^2\ core\_size(G_\omega))$ time. Thus, the total time for level i, obtained by summing over all level $i-1$ graphs G_ω, is $O(c^2 n)$. Summed over all levels i, this is $O(c^2 n \log n)$.

If G_ω is a graph with at most c core nodes, then milestone information can be set up as above for any pair of core nodes. The time per pair is $O(c)$, hence $O(c^3)$ for all pairs. Since there are $O(n/c)$ such graphs G_ω, the total time for all graphs G_ω is $O(c^2 n)$.

It follows that the total setup time for the basic scheme is $O(cn(\log n)^2 + c^4 n \log n)$. For planar graphs, the faster algorithm from [F] may be used in lieu of

the algorithm from [J] for determining shortest paths. This leads to a setup time of $O(cn(\log n)^{3/2} + c^2 n \log n + c^4 n(\log n)^{1/2})$. For series-parallel graphs, a setup time of $O(n \log n)$ can be achieved, using a result from [HT] which allows single-source shortest paths to be computed in $O(n)$ time.

The analysis for the improved scheme is as follows. The time for doing the decomposition and naming, and for setting up shortest paths information, milestones and surrogates is as before. The time to encode the additional information into the name of a level i node v is as follows. A lexicographic ordering of all the nodes, based on the names assigned from the decomposition, can be generated in $O(n \log n)$ time using a radix sort. From this ordering, a lexicographic ordering of of the ancestors of v for any level $j < i$ can be inferred in $O(c)$ time. Since the distances from v to the ancestors are known, the distance ordering can be generated and the corresponding information encoded into v's name in $O(c \log c)$ time. Furthermore, given $\alpha(c)$, the information about the relative magnitudes of distances can be determined in $O(c)$ time, and encoded in v's name in $O(c \log c)$ time. Thus, the time per level for v is $O(c \log c)$, hence $O(c \log c \log n)$ for all levels. Taken over all nodes this is $O((c \log c) n \log n)$.

Thus the overall setup time is $O(cn(\log n)^2 + c^4 n \log n)$. It is $O(cn(\log n)^{3/2} + c^2 n \log n + c^4 n(\log n)^{1/2})$ for planar networks, and $O((c \log c) n \log n)$ for series-parallel networks.

We thus have the following theorem.

Theorem 6. Let G be any graph from a uniformly sparse and contractable class of c-decomposable graphs with a linear-time c-separator algorithm. The basic and improved routing schemes can be implemented in G in $O(cn(\log n)^2 + c^4 n \log n)$

time. If G is planar then the time is $O(cn(\log n)^{3/2} + c^2 n \log n + c^4 n(\log n)^{1/2})$. ∎

References

[B] B.S. Baker, "Approximation algorithms for NP-complete problems on planar graphs", *Proceedings of the* 24^{th} *IEEE Annual Symposium on Foundations of Computer Science*, 1983, pp. 265–273.

[D] R.J. Duffin, "Topology of series-parallel networks", *Journal of Mathematical Applications*, Vol. 10, No. 2, 1965, pp. 303–318.

[F] G.N. Frederickson, "Fast algorithms for shortest paths in planar graphs, with applications", CSD-TR-486 (revised), Purdue University, April 1985. (An earlier version appeared as: "Shortest path problems in planar graphs", *Proceedings of the* 24^{th} *Annual IEEE Symposium on Foundations of Computer Science*, 1983, pp. 242–247.)

[FJ1] G.N. Frederickson and R. Janardan, Optimal message routing without complete routing tables, *Proceedings of the* 5^{th} *Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, Calgary, August 1986, pp. 88–97.

[FJ2] G.N. Frederickson and R. Janardan, Efficient message routing in planar networks, CSD-TR-638, Purdue University, November 1986.

[FJ3] G.N. Frederickson and R. Janardan, "Separator-based strategies for efficient message routing", *Proceedings of the* 27^{th} *Annual IEEE Symposium on Foundations of Computer Science*, Toronto, October 1986, pp. 428–437.

[H] F. Harary, *Graph Theory*, Addison-Wesley, Reading MA, 1969.

[HT] R. Hassin and A. Tamir, "Efficient algorithms for optimization and selection on series-parallel graphs", *SIAM Journal on Algebraic and Discrete Methods*, Vol. 7, No. 3, July 1986, pp. 379–389.

[J] D.B. Johnson, "Efficient algorithms for shortest paths in sparse networks", *Journal of the ACM*, Vol. 24, No. 1, January 1977, pp. 1–13.

[LT] R.J. Lipton and R.E. Tarjan, "A separator theorem for planar graphs", *SIAM Journal on Applied Mathematics*, Vol. 36, No. 2, April 1979, pp. 177–189.

[M] G. Miller, "Finding small simple cycle separators for 2-connected planar graphs", *Proceedings of the* 16^{th} *Annual ACM Symposium on Theory of Computing*, 1984, pp. 376–382.

[SK] N. Santoro and R. Khatib, "Labelling and implicit routing in networks", *The Computer Journal*, Vol. 28, No. 1, February 1985, pp. 5–8.

[vLT1] J. van Leeuwen and R.B. Tan, "Routing with compact routing tables", Technical Report RUU-CS-83-16, Department of Computer Science, University of Utrecht, The Netherlands, November 1983.

[vLT2] J. van Leeuwen and R.B. Tan, "Interval routing", Technical Report RUU-CS-85-16, Department of Computer Science, University of Utrecht, The Netherlands, May 1985.

[VTL] J. Valdes, R.E. Tarjan and E.L. Lawler, "The recognition of series-parallel digraphs", *SIAM Journal on Computing*, Vol. 11, No. 2, May 1982, pp. 298–313.

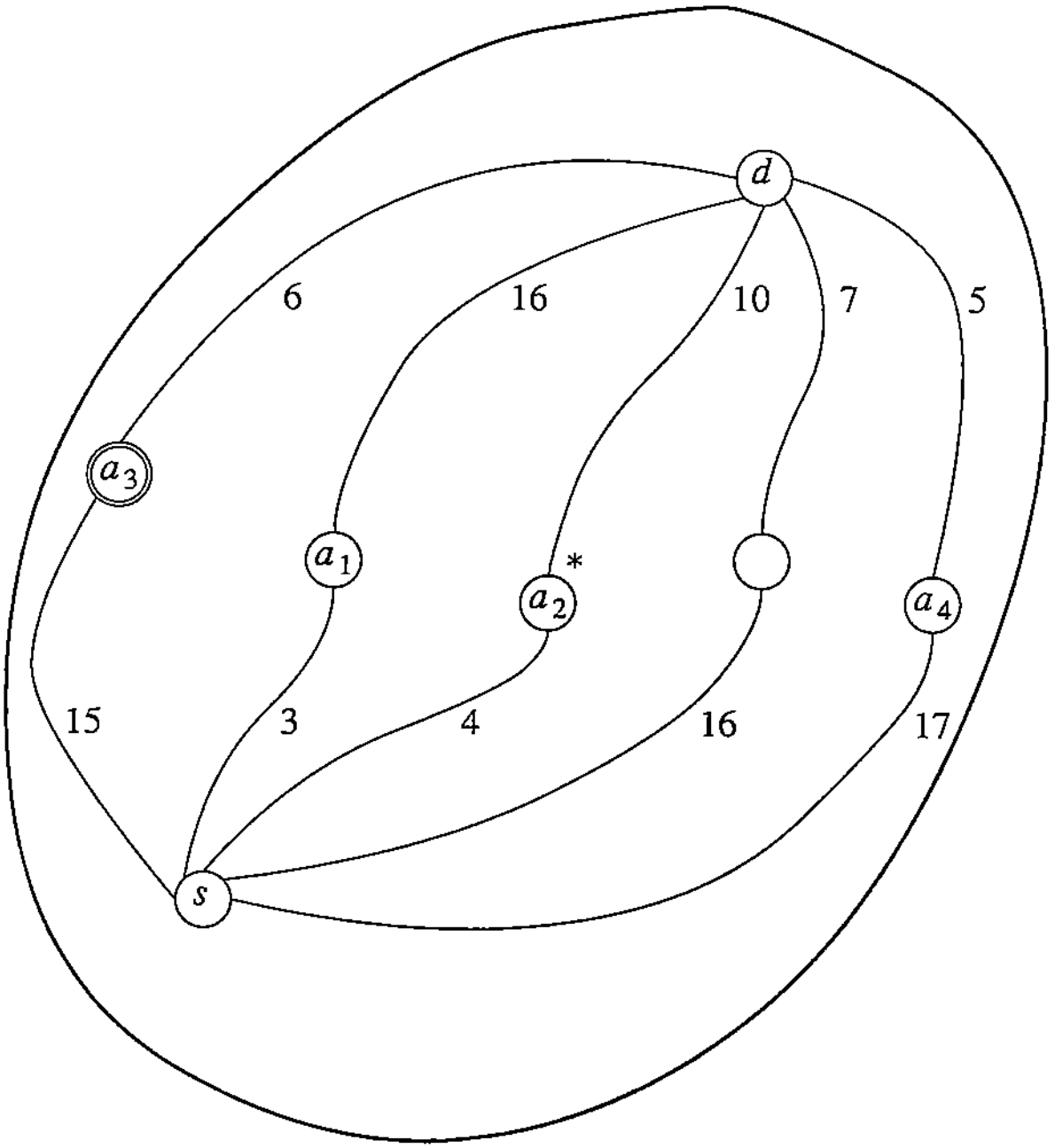

Figure 1. Example of a routing choice in the improved scheme for a c-decomposable network, with $c = 5$.