SIAM J. Discrete Math.
Vol. 23, No. 4, pp. 2079–2092

IMPROVED COMPACT ROUTING TABLES FOR PLANAR NETWORKS VIA ORDERLY SPANNING TREES[*]

HSUEH-I LU[†]

Abstract. We address the problem of designing compact routing tables for an unlabeled connected n-node planar network G. For each node r of G, the designer is given a routing spanning tree T_r of G rooted at r, which specifies the routes for sending packets from r to the rest of G. Each node r of G is equipped with ports $1, 2, \ldots, deg_r$, where deg_r is the degree of r in T_r. Each port of r is supposed to be assigned to a neighbor of r in T_r in a one-to-one manner. For each node v of G with $v \neq r$, let $port_r(v)$ be the port to which r should forward packets with destination v. Under the assumption that the designer has the freedom to determine the label and the port assignment of each node in G, the *routing table design problem* is to design a compact routing table R_r for each node r such that $port_r(v)$ can be determined merely from R_r and the label of v. Compact routing tables for various network topologies have been extensively studied in the literature. Planar networks are particularly important for routing with geometric metrics. Based upon four-page decompositions of G, Gavoille and Hanusse gave the best previously known polynomial-time computable result for this problem with linear-space routing tables, where the time complexity is measured under the conventional unit-cost RAM model of computation:

- Each $port_r(v)$ is computable from R_r and the label of v in $O(\log^{2+\epsilon} n)$ time for any positive constant ϵ.
- The number of bits required to encode each R_r is at most $8n + o(n)$.
- The time required to compute each R_r is $O(n)$.

Based on orderly spanning trees of G, our design achieves the following improved bounds without increasing the time complexity for computing each R_r:

- Each $port_r(v)$ is computable from R_r and the label of v in $O(\log^{1+\epsilon} n)$ time for any positive constant ϵ.
- The number of bits required to encode each R_r is at most $7.181n + o(n)$.
- The overall code length of all n routing tables is at most $7n^2 + o(n^2)$ bits.

Key words. planar network, routing table, port assignment, graph encoding, orderly spanning tree, succinct data structure, unit-cost RAM model

AMS subject classifications. 68M10, 68P05, 68Q25, 68R05, 68W05, 68W40, 92-08

DOI. 10.1137/070703041

1. Introduction.

All graphs in the paper have no multiple edges or self-loops. All logarithms are taken to base two. Throughout the paper, we stick with the conventional $O(\log n)$-bit unit-cost RAM model of computation [10, 24, 31, 48, 51, 64], which assumes that operations read, write, and add on $O(\log n)$ consecutive bits take $O(1)$ time.

We address the problem of designing compact routing tables for an unlabeled connected planar network G over an n-node set V. For each node r of G, we are given

[*]Received by the editors September 16, 2007; accepted for publication (in revised form) September 23, 2009; published electronically January 15, 2010. A preliminary version of this paper appeared in *COCOON* 2002 [47]. The result in the preliminary version is further improved by the current version in the following aspects: (a) Due to a recent result of Lu and Yeh [48], the construction for routing tables in the current version is more elegant. (b) It takes expected linear time to construct each routing table in the preliminary version. The construction in the current version takes worst-case linear time. (c) The current version ensures a tighter bound on the overall code length.
http://www.siam.org/journals/sidma/23-4/70304.html

[†]Department of Computer Science and Information Engineering, National Taiwan University, 1 Roosevelt Road, Section 4, Taipei 106, Taiwan, Republic of China (hil@csie.ntu.edu.tw, www.csie.ntu.edu.tw/~hil/). Research was supported in part by NSC grants 96-2221-E-002-033, 97-2221-E-002-122, and 98-2221-E-002-079-MY3. The author also holds joint appointments in the Graduate Institute of Networking and Multimedia and the Graduate Institute of Biomedical Electronics and Bioinformatics, National Taiwan University.

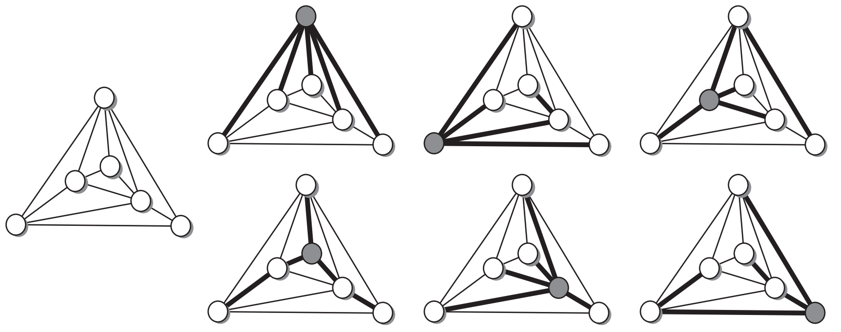

FIG. 1. *A six-node planar network and six routing spanning trees rooted at the gray nodes.*

a *routing spanning tree* T_r of G rooted at r, which specifies the routes for sending packets from r to the rest of G. For instance, these routing spanning trees could be carefully designed to avoid high congestion of the network packets in G. T_r could also be a precomputed shortest-path tree rooted at r for some weighted version of G. An example is shown in Figure 1. Although not required by our result, it is reasonable to assume that those n routing spanning trees are *consistent*; i.e., if s and v are two nodes such that s is on the path of T_r between r and v, then the path of T_s between s and v is identical to the path of T_r between s and v. Each node r of G is equipped with ports $1, 2, \ldots, deg_r$, where deg_r is the degree of r in T_r. Each port of r is supposed to be assigned to a neighbor of r in T_r in a one-to-one manner. The routing table designer has the freedom to determine the label and the port assignment of each node in G. For each neighbor s of r in T_r, let $port_r(s)$ denote the index of the port of r that is assigned to s. For each node v of G with $v \neq r$ that is not a neighbor of r in T_r, define $port_r(v) = port_r(s)$, where s is the neighbor of r in T_r whose removal disconnects v and r in T_r. That is, if r receives a packet whose destination is v, then r should pass the packet to its port with index $port_r(v)$. The *routing table design problem* is to come up with a compact routing table R_r for each node r of G such that $port_r(v)$ can be determined merely from R_r and the label of v. Natural objectives of this problem include:

- minimizing the number $\lambda_r(n)$ of bits to encode the label of node r;
- minimizing the time $\tau_r(n)$ required to obtain $port_r(v)$ from R_r and the label of v;
- minimizing the time $\pi_r(n)$ required to compute R_r from G;
- minimizing the number $\rho_r(n)$ of bits to encode R_r; and
- minimizing the overall code length $\sum_{r\in V} \rho_r(n)$.

For instance, a straightforward solution, which also works for non-planar G, achieves the following bounds: $\lambda_r(n) = \lceil \log n \rceil$, $\tau_r(n) = O(1)$, $\pi_r(n) = O(n)$, $\rho_r(n) = n \cdot \lceil \log n \rceil$. However, $\rho_r(n)$ can be reduced by exploiting the planarity of G. Based upon four-page decompositions of G [65], Gavoille and Hanusse [28] gave the best formerly known result for this problem with linear $\rho_r(n)$ and polynomial $\tau_r(n)$, which outperformed several previous trade-offs among the above objectives [12, 35, 63]. Specifically, their design achieves the following bounds for each node r:

$$\begin{aligned} \lambda_r(n) &= \lceil \log n \rceil, \\ \pi_r(n) &= O(n), \\ \tau_r(n) &= O\left(\log^{2+\epsilon} n\right), \\ \rho_r(n) &\le 8n + o(n), \end{aligned}$$

where ϵ can be any positive constant. Gavoille and Hanusse [28] also mentioned a lower bound $\rho_r(n) \geq n - O(\log n)$ for the routing table design problem on G with $\lambda_r(n) = \lceil \log n \rceil$. While maintaining the same bounds on $\lambda_r(n)$ and $\pi_r(n)$, we obtain the following improved bounds on query time and code length:

$$\begin{aligned} \tau_r(n) &= O\left(\log^{1+\epsilon} n\right), \\ \rho_r(n) &< 7.181n + o(n), \\ \sum_{r \in V} \rho_r(n) &\leq 7n^2 + o(n^2), \end{aligned}$$

where ϵ can be any positive constant. Moreover, our design has an additional feature that T_r can be recovered from R_r in $O(n)$ time.

Compactness of routing tables for various network topologies has been extensively studied in the literature [1, 2, 13, 14, 16–18, 20–22, 26, 27, 29, 30, 41, 42, 55, 56, 62]. Planar networks are particularly important in routing with geometric metrics [6, 34, 40, 49, 50, 59]. For example, some of the best network topology maps used by ISPs and Internet Backbone Networks can be modeled as planar or almost-planar graphs [25, 39, 43]. The near-shortest routes in wireless ad hoc networks are obtained through various types of planar subnetworks with low stretch factors [7, 8, 43, 53]. If T_r is restricted to routing spanning trees with $(1 + \epsilon)$-stretch, Thorup [60, 61] gave a solution with poly-logarithmic $\rho_r(n)$. Other results related to routing in planar networks can be found in [3, 5, 23, 46].

Our improved compact routing tables are based upon succinct encodings for planar graphs with efficient query support [9, 10, 12, 36, 51]. The best of such results for an n-node m-edge planar graph, due to Chiang, Lin, and Lu [10], takes $2m + 2n + o(n)$ bits, is obtainable in $O(n)$ time, and supports $O(1)$-time adjacency and degree queries. Augmented with a recent result of Lu and Yeh [48], the encoding also supports $O(1)$-time query for selecting the ith neighbor of any node, as to be shown in Lemma 5. Chiang et al.'s encoding is based on an algorithmic tool called *orderly spanning tree*, which generalizes the concepts of realizers [52, 57, 58] and canonical orderings [11, 15, 33, 37, 38] for planar graphs. Besides graph encoding, orderly spanning tree also finds applications in graph drawing [10] and VLSI floor planning [44, 45]. Our improved design of routing tables relies heavily on Chiang et al.'s techniques for encoding planar graphs via orderly spanning trees. Therefore, our results can be regarded as an application of orderly spanning trees in computer networks.

The rest of the paper is organized as follows. Section 2 gives the preliminaries. Section 3 describes our design of routing tables. Section 4 shows how to efficiently obtain the correct port from the routing table and the label of the destination. Section 5 concludes the paper with a couple of open questions.

2. Preliminaries.

For any string X, let $|X|$ denote the number of bits to encode X.

LEMMA 1 (see [4, 19]). *Let $B_1, \ldots, B_k$ be binary strings. If $\sum_{i=1}^{k} |B_i| = n^{O(1)}$ and $k = O(1)$, then there exists an $o(n)$-bit string β, obtainable in $O(n)$ time, such that given the concatenation of $\beta, B_1, B_2, \ldots, B_k$, the position of the first bit of each B_i in the concatenation can be computed in $O(1)$ time.*

Let $B_1 \circ B_2 \circ \cdots \circ B_k$ denote the concatenation of $\beta, B_1, B_2, \ldots, B_k$ as in Lemma 1.

2.1. Known succinct dictionaries.

For any string X, let $X[i]$ denote the ith character of X. Let e denote the base of the natural logarithm.

Lemma 2 (Hagerup, Miltersen, and Pagh [32]). *Let B be an n-bit binary string with exactly k one bits. It takes $O(k \log k)$ time to encode B into a string $Z_1(B)$ with $|Z_1(B)| = O(k \log n)$ such that each $B[i]$ can be determined from $Z_1(B)$ in $O(1)$ time.*

Lemma 3 (Pagh [54]). *Let B be an n-bit binary string with exactly k one bits, where $n = k\,(\log k)^{O(1)}$. It takes $O(n)$ time to encode B into a binary string $Z_2(B)$ with $|Z_2(B)| = k \log \frac{en}{k} + o(n)$ such that each $B[i]$ can be determined from $Z_2(B)$ in $O(1)$ time.*

2.2. Succinct representations of binary strings with rank support. For notational brevity, let $\tilde{O}(t)$ denote $O(t) \cdot O((\log\log n)^{O(1)})$. Define

$$L = \lceil (\log n)(\log\log n) \rceil,$$
$$\ell = \lceil (\log\log n)^2 \rceil.$$

Clearly, $L = \tilde{O}(\log n)$ and $\ell = \tilde{O}(1)$. For any binary string B, let $B\langle i\rangle$ denote the ith $\lceil \log n\rceil$-bit word of B; let $B\{i\}$ denote the ith $\lceil \log L\rceil$-bit word of B; and let $\mathit{rank}(B, i)$ denote the number of one bits in $B[1], B[2], \ldots, B[i]$.

Lemma 4. *Let B be an n-bit binary string with at most k one bits. It takes $O(n)$ time to encode B into a string $Z(B)$ with*

$$|Z(B)| \le \begin{cases} o(n) & \text{if } k = o(n), \\ \min\left(n, k \log \frac{en}{k}\right) + o(n) & \text{if } k = \Omega(n) \end{cases}$$

such that each $B[i]$ is obtainable from $Z(B)$ in $O(1)$ time. Moreover, each $\mathit{rank}(B, i)$ can be determined from $Z(B)$ in $\tilde{O}(1)$ time.

Proof. Let k' be the number of one bits in B. We have $k' \le k$. Let $Z(B) = Z \circ Z' \circ Z''$, where Z, Z', and Z'' are defined as follows:

- The content of Z depends on whether $k' \le \frac{n}{L}$.
  - If $k' \le \frac{n}{L}$, then let $Z = Z_1(B)$. By Lemma 2, Z can be computed from B in $O(k' \log k') = O(n)$ time. We have $|Z| = O(k' \log n) = o(n)$. Each $B[i]$ can be computed from Z in $O(1)$ time.
  - If $k' > \frac{n}{L}$, then $n = k'(\log k')^{O(1)}$. Let Z be the shorter of B and $Z_2(B)$. By Lemma 3, Z can be computed from B in $O(n)$ time. We have $|Z| = \min(n, k' \log \frac{en}{k'}) + o(n)$. Each $B[i]$ can be computed from Z in $O(1)$ time.

  If $k = o(n)$, then $k' = o(n)$, thereby $|Z| = |Z_1(B)| = o(n)$. As for the case with $k = \Omega(n)$, define $f(x) = x \log \frac{en}{x}$. Observe that $f(x)$ is monotonically increasing for $0 \le x \le \frac{n}{e}$. Since $f(x) \ge n$ holds for $\frac{n}{e} \le x \le n$, $\min(n, f(x))$ is monotonically increasing for $0 \le x \le n$. It follows that $|Z| = |Z_2(B)| = \min(n, f(k')) + o(n) \le \min(n, f(k)) + o(n)$.
- Let Z' be the binary string such that $Z'\langle j_1\rangle = \mathit{rank}(B, j_1 \cdot L)$ for each $j_1 = 0, 1, \ldots, \lfloor \frac{n}{L} \rfloor$. Clearly, Z' can be computed from B in $O(n)$ time and $|Z'| = o(n)$.
- Let Z'' be the binary string such that $Z''\{j_2\}$ keeps the number of one bits from $B[L \cdot \lfloor j_2 \cdot \ell / L\rfloor + 1]$ to $B[j_2 \cdot \ell]$, for each $j_2 = 0, 1, \ldots, \lfloor \frac{n}{\ell} \rfloor$. Clearly, Z'' can be computed from B in $O(n)$ time and $|Z''| = o(n)$.

Since $|Z'| + |Z''| = o(n)$, one can verify that the bounds on $|Z(B)|$ hold for both cases of k. For each index i, it takes $\tilde{O}(1)$ time to determine $\mathit{rank}(B, i)$ from $Z(B)$ as follows: Let $j_1 = \lfloor \frac{i}{L} \rfloor$ and $j_2 = \lfloor \frac{i}{\ell} \rfloor$. For each index $j = j_2 \cdot \ell + 1, j_2 \cdot \ell + 2, \ldots, i$, we obtain $B[j]$ from Z in $O(1)$ time. As a result, we know the number j_3 of one

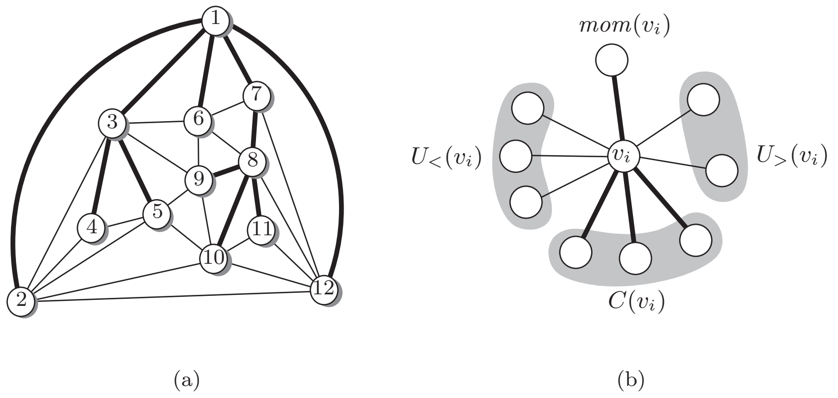

FIG. 2. (a) *A plane graph G with an orderly spanning tree T of G rooted at node* 1*. The node labels show the counterclockwise preordering of the nodes in T.* (b) *Illustration for a node being orderly.*

bits from $B[j_2 \cdot \ell + 1]$ to $B[i]$ in $\tilde{O}(1)$ time. We have $rank(B, i) = Z'\langle j_1 \rangle + Z''\{j_2\} + j_3$. $\square$

2.3. Succinct encodings for planar graphs via orderly spanning trees. Let T be a rooted spanning tree of a plane graph G. Two nodes are *unrelated* in T if they are distinct and neither of them is an ancestor of the other in T. Let $v_1, v_2, \ldots, v_n$ be the counterclockwise preordering of the nodes in T. As illustrated in Figure 2(b), node v_i is *orderly* in G with respect to T if the neighbors of v_i in G form the following four blocks in counterclockwise order around v_i, where each block could be empty:

- the parent $mom(v_i)$ of v_i in T;
- $U_<(v_i)$ consisting of the unrelated neighbors v_j of v_i in G with $j < i$;
- $C(v_i)$ consisting of the children of v_i in T; and
- $U_>(v_i)$ consisting of the unrelated neighbors v_j of v_i in G with $j > i$.

T is an *orderly spanning tree* of G if v_1 is on the boundary of the exterior face of G, and each $v_i, 1 \le i \le n$ is orderly in G with respect to T. An example of an orderly spanning tree is shown in Figure 2(a). For any connected planar graph H, Chiang et al. [10] gave a linear-time algorithm that computes a plane embedding G of H and an orderly spanning tree of G.

Let us review the details of Chiang et al.'s encoding [10] that are relevant to our improved routing tables. The encoding can be constructed via any orderly spanning tree T of any plane embedding G of the input n-node m-edge planar graph. For each $i = 1, 2, \ldots, n$, let v_i be the ith node in the counterclockwise preordering of the nodes in T. Chiang et al.'s encoding for G has $2n + 2m + o(n)$ bits, whose major piece is a compact representation for a string P consisting of $2n$ parentheses and $2m - 2n + 2$ brackets. Specifically, the parentheses (respectively, brackets) are balanced in P; i.e., each of them has a matching parenthesis (respectively, bracket) in P.

- Each matching parenthesis pair in P corresponds to a node in G. For each $i = 1, 2, \ldots, n$, let $(_i$ denote the ith open parenthesis in P; and let $)_i$ denote the parenthesis that matches $(_i$ in P. The matching $(_i$ and $)_i$ is the pair corresponding to v_i. Moreover, v_i is an ancestor of v_j in T if and only if $(_i$ and $)_i$ enclose $(_j$ and $)_j$ in P.
- Each matching bracket pair in P corresponds to an edge in $G - T$. For each $k = 1, 2, \ldots, m - n + 1$, let $[_k$ be the kth open bracket in P, and $]_k$ its

matching bracket in P. The matching pair $\texttt{[}_k$ and $\texttt{]}_k$ corresponds to the edge $e_k = (v_i, v_j)$, where $\texttt{)}_i$ (respectively, $\texttt{(}_j$) is the rightmost parenthesis that precedes $\texttt{[}_k$ (respectively, $\texttt{]}_k$) in P. We call k the *identifier* of (v_i, v_j) and write $id(v_i, v_j) = k$.

An explicit representation of P requires $4m + O(1)$ bits, since P has length $2m + O(1)$ and consists of four distinct symbols. Since the direction of a bracket can be determined from that of its rightmost preceding parenthesis, P can be encoded into two strings S' and S'' with $|S'| + |S''| = 2m + 2n + O(1)$:

- S' has $2n$ bits, signifying whether each of the $2n$ parentheses is open or not.
- S'' has $2m + O(1)$ bits, signifying whether $P[i]$ is a parenthesis or a bracket.

For example, if G and T are as shown in Figure 2(a), then we have

$$P = \texttt{(()[[[[[(](])[(]])[[)[[(])[[[(](](]]])[(]]])[[(])[)[)[(]]]]]))},$$
$$S' = \texttt{(()(()())()((()()()))())},$$
$$S'' = \texttt{1110000010101010010010010100010101000101000100101010101000011}.$$

Augmenting S' and S'' with some $o(n)$-bit auxiliary strings provided by Chiang, Lin, and Lu [10] and Lu and Yeh [48], many queries on G and T can be answered in $O(1)$ time. Let $nbr(v_1, j)$ denote the jth neighbor of v_1 in G in counterclockwise order around v_1 starting from v_2. If $i > 1$, then let $nbr(v_i, j)$ denote the jth neighbor of v_i in G in counterclockwise order around v_i starting from $mom(v_i)$. For instance, if G and T are as shown in Figure 2(a), then we have $nbr(v_1, 3) = v_6$, $nbr(v_3, 1) = v_1$, and $nbr(v_8, 6) = v_{12}$. We have the following lemma.

LEMMA 5. *There is an $O(n)$-time computable $2m + 2n + o(n)$-bit encoding S for G and T from which each of the following queries can be answered in $O(1)$ time for any indices i and j:*

- *whether nodes v_i and v_j are adjacent in G;*
- *whether nodes v_i and v_j are adjacent in T;*
- *$id(v_i, v_j)$ for any edge $(v_i, v_j) \in G - T$; and*
- *$nbr(v_i, j)$.*

Proof. Theorem 5.4 of Chiang et al. [10] shows an $o(n)$-bit string σ' such that the first three queries in the lemma can be answered in $O(1)$ time from $S' \circ S'' \circ \sigma'$. To answer $nbr(v_i, j)$ in $O(1)$ time, observe that Theorem 5.4 of Chiang et al. [10] also shows that the following three queries can be answered from $S' \circ S'' \circ \sigma'$ in $O(1)$ time for any indices i and j:

- the numbers of neighbors of v_i in $U_<(v_i)$, $C(v_i)$, and $U_>(v_i)$;
- the jth neighbor of v_i in $U_<(v_i)$ in counterclockwise order around v_i; and
- the jth neighbor of v_i in $U_>(v_i)$ in counterclockwise order around v_i.

Moreover, Lu and Yeh [48] gave an $o(n)$-bit string σ'' such that the jth neighbor of v_i in $C(v_i)$ in counterclockwise order around v_i can be identified from $S' \circ \sigma''$ in $O(1)$ time. Putting the above observations together, one can see that it takes $O(1)$ time to determine $nbr(v_i, j)$ from $S = S' \circ S'' \circ \sigma' \circ \sigma''$. $\square$

3. The improved design of routing tables.

Without loss of generality, we may assume that the input n-node planar graph G is a plane triangulation. By Theorem 3.2 of Chiang et al. [10], it takes $O(n)$ time to obtain three orderly spanning trees T^a, T^b, and T^c of G such that each edge of G appears in at least one of T^a, T^b, and T^c. For instance, Figure 3 gives two other orderly spanning trees of the plane triangulation shown in Figure 2(a). For each $T \in \{T^a, T^b, T^c\}$, we obtain a design of routing tables with respect to T. The final design to output is the one whose $\sum_{r \in V} |R_r|$ is minimum among the three.

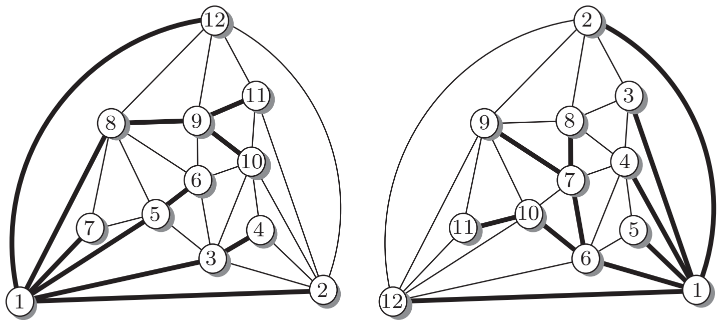

FIG. 3. *Two other orderly spanning trees of the plane triangulation shown in Figure* 2(a). *The node labels show the counterclockwise preordering of the corresponding orderly spanning trees.*

Given any orderly spanning tree T of G, our design of routing tables with respect to T is as follows. Since the edges of G not in $T \cup T_r$ are irrelevant to the information to be stored in R_r, it suffices to focus on the plane graph $G_r = T \cup T_r$. Let m_r be the number of edges in G_r. Observe that $m_r \leq 2n - 2$. Both T and T_r are spanning trees of G_r. Moreover, T remains an orderly spanning tree of G_r.

Let $v_1, v_2, \ldots, v_n$ be the counterclockwise preordering of the nodes in T. For each $i = 1, 2, \ldots, n$, let $i-1$ be the label of v_i, which can be encoded in $\lceil \log n \rceil$ bits. As for the port assignment, for each $j = 1, 2, \ldots, deg_r$, we simply assign port j of r to the neighbor of r in T_r, with the jth smallest label. It remains to describe the content of R_r for each node r.

Let $dad_r(v_i)$ denote the parent of v_i in T_r. An edge (v_i, v_j) of T_r with $i < j$ is *forward* (respectively, *backward*) in G_r if $v_i = dad_r(v_j)$ (respectively, $v_j = dad_r(v_i)$). Each edge in $G_r - T$ is either forward or backward. However, an edge of T may be neither forward nor backward, since an edge of T may not be in T_r.

A node is *popular* if it has at least L neighbors in G_r. A node is *lonely* if it has at most ℓ neighbors in G_r. Since G_r has $O(n)$ edges, the number of popular nodes is $O(\frac{n}{L})$ and the number of non-lonely nodes is $O(\frac{n}{\ell})$.

We choose $O(\frac{n}{L})$ *special* nodes from T_r as follows. Choose an arbitrary leaf u with maximum depth in T_r. Let w be the highest ancestor of u in T_r whose distance from u in T_r is no more than L. Observe that if $w \neq r$, then w is the ancestor of u whose distance from u in T_r is exactly L. We then choose w as a special node, delete the subtree of T_r rooted at w from T_r, and then repeat the above steps until all nodes are deleted from T_r. By the above choice of special nodes, each node u in T_r either has depth less than L or has a special ancestor whose distance from u is at most L. Let

$$R_r = S_1 \circ S_2 \circ S_3 \circ S_4 \circ S_5 \circ S_6 \circ S_7,$$

where $S_1, \ldots, S_7$ are defined below. The idea behind these seven strings is as follows:

- S_1 encodes G and T, which form the basis of our encoding. The node labeling of G is simply the counterclockwise preordering of T. We use the natural port assignment; i.e., the ith port of node r is reserved for the neighbor of r in T_r with the ith smallest label. Node r should forward a packet with destination z to the neighbor, say y, of r in T_r such that the subtree of T_r rooted at y contains z. Therefore, node y can be derived from node z via the parent queries for T_r, which is supported by S_2, S_3, S_4, and S_5. Once we know the label of node y, we use S_7 to figure out the corresponding port number.

- S_6 is to remedy the problem that the distance between y and z in T_r could be large. That is, the number of parent queries in T_r required to derive y from z could be $\Theta(n)$. To reduce the number of required queries, we select $o(\frac{n}{\log n})$ special nodes such that if we traverse in T_r from node z towards node r, in $O(\log n \log\log n)$ steps we reach either node y or a special node that is an ancestor of z in T_r whose distance from z in T_r is $O(\log n \log\log n)$. Since the number of special nodes is $o(\frac{n}{\log n})$, we can afford to spend $O(\log n)$ bits for each special node to encode the port number assigned to y.
- S_2 and S_3 together store T_r by encoding the direction of each edge of G in T_r. As a result, we can determine the parent of a lonely node in T_r in $O(\log n \log\log n)$ time by checking the directions of its $O(\log n \log\log n)$ incident edges.
- S_4 takes care of popular nodes with $o(n)$ bits. Since the number of popular nodes is $o(\frac{n}{\log n})$, we can afford to spend $O(\log n)$ bits for each popular node to indicate which of its neighbors is its parent in T_r.
- S_5 takes care of nodes that are neither popular nor lonely with $o(n)$ bits. Since the number of non-lonely nodes is $o(\frac{n}{\log\log n})$, we can afford to spend $O(\log\log n)$ bits for each non-lonely node. Since each non-popular node has $O(\log n \log\log n)$ neighbors, $O(\log\log n)$ bits suffice to indicate which of its neighbors is its parent in T_r.

The detailed description for $S_1, \ldots, S_7$ is as follows. We will see that $|S_4| + |S_5| + \cdots + |S_7| = o(n)$. Therefore, only S_1, S_2, and S_3 may contribute to the first-order term of $|R_r|$.

1. Let S_1 be the $O(n)$-time computable encoding for G_r and T with $|S_1| = 2m_r + 2n + o(n)$ as stated in Lemma 5.
2. Let S_2 encode the direction of each edge in $G_r - T$. Specifically, let S_2 be the $(m_r - n + 1)$-bit binary string such that $S_2[i] = 1$ if and only if the edge of $G_r - T$ with identifier i is forward. S_2 can be computed from G_r, T, and T_r in $O(n)$ time. It takes $O(1)$ time to obtain each $S_2[i]$ from S_2.
3. Let $S_3 = Z(S'_3)$, where S'_3 encodes whether each edge of T is a backward edge of T_r. Specifically, let S'_3 be the n-bit binary string such that $S'_3[i] = 1$ if and only if edge $(v_i, mom(v_i))$ is a backward edge of T_r.

   By Lemma 4, it takes $O(1)$ time to obtain each $S'_3[i]$ from S_3. Since the number of one bits in S'_3 is at most $2n - m_r$. By Lemma 4, S_3 can be computed from G_r, T, and T_r in $O(n)$ time, and
   $$|S_3| \le \min\left(n, (2n - m_r)\log\frac{en}{2n - m_r}\right) + o(n).$$
4. Let $S_4 = Z(S'_4) \circ S''_4$, where S'_4 encodes whether each node is popular, and S''_4 stores the parent of each popular node in T_r. Specifically, let S'_4 be the n-bit binary string such that $S'_4[i] = 1$ if and only if v_i is popular. Since the number of popular nodes is $O(\frac{n}{L})$, the number of one bits in S'_4 is $o(\frac{n}{\log n})$. By Lemma 4, we have $|Z(S'_4)| = o(n)$. Let S''_4 be the binary string such that $S''_4\langle j\rangle$ stores the label of the parent of the jth popular node in T_r. Since the number of popular nodes is $o(\frac{n}{\log n})$, we have $|S''_4| = o(n)$, thereby $|S_4| = o(n)$. By Lemma 4, S_4 can be computed from G_r, T, and T_r in $O(n)$ time.

   By Lemma 4, it takes $O(1)$ time to determine whether v_i is popular by fetching $S'_4[i]$ from $Z(S'_4)$. If $S'_4[i] = 1$, then $dad_r(v_i) = v_j$ with $j = S''_4\langle rank(S'_4, i)\rangle$, which by Lemma 4 can be obtained from $Z(S'_4)$ and S''_4 in $\tilde{O}(1)$ time.

5. Let $S_5 = Z(S'_5) \circ S''_5$, where S'_5 encodes whether each node is neither popular nor lonely and S''_5 stores the position of $dad_r(v)$ among the neighbors of v in G_r for each non-popular and non-lonely node v. Specifically, let S'_5 be the n-bit binary string such that $S'_5[i] = 1$ if and only if v_i is neither lonely nor popular. Since the number of non-lonely nodes is $O(\frac{n}{\ell})$, the number of one bits in S'_5 is $o(\frac{n}{\log\log n})$. By Lemma 4, we have $|Z(S'_5)| = o(n)$. Let S''_5 be the binary string such that if $S'_5[i]$ is the jth one bit in S'_5 (i.e., $S'_5[i] = 1$ and $rank(S'_5, i) = j$), then $S''_5\{j\}$ keeps the index k_j with $dad_r(v_i) = nbr(v_i, k_j)$. Observe that if $S'_5[i] = 1$, then v_i is not popular. Therefore, $k_j \le L$ can be encoded in $\lceil \log L \rceil = O(\log\log n)$ bits. Since the number of non-lonely nodes is $o(\frac{n}{\log\log n})$, we have $|S''_5| = o(n)$, thereby $|S_5| = o(n)$. By Lemma 4, S_5 can be computed from G_r, T, and T_r in $O(n)$ time.

   By Lemma 4, it takes $O(1)$ time to determine whether v_i is neither popular nor lonely by fetching $S'_5[i]$ from $Z(S'_5)$. If $S'_5[i] = 1$, then we know that $dad_r(v_i)$ is the neighbor v_j of v_i with $j = S''_5\{rank(S'_5, i)\}$. By Lemma 4, $S''_5\{rank(S'_5, i)\}$ can be obtained from $Z(S'_5)$ and S''_5 in $\tilde{O}(1)$ time. It follows from Lemma 5 that $nbr(v_i, j)$ can be obtained from S_1 in $O(1)$ time. To sum up, if v_i is neither popular nor lonely, it takes $\tilde{O}(1)$ time to obtain $dad_r(v_i)$ from S_1 and S_5.
6. Let $S_6 = Z(S'_6) \circ S''_6$, where S'_6 encodes whether each node is special, and S''_6 stores $port_r(v)$ for each special node v. Specifically, let S'_6 be the n-bit binary string such that $S'_6[i] = 1$ if and only if v_i is special. Since the number of special nodes is $O(\frac{n}{L})$, the number of one bits in S'_6 is $o(\frac{n}{\log n})$. By Lemma 4, we have $|Z(S'_6)| = o(n)$. Let S''_6 be the binary string such that $S''_6\langle j \rangle$ stores $port_r(v_i)$, where v_i is the jth special node in G; i.e., $S'_6[i] = 1$ and $rank(S'_6, i) = j$. Since the number of special nodes is $o(\frac{n}{\log n})$, we have $|S''_6| = o(n)$, thereby $|S_6| = o(n)$. By Lemma 4, S_6 can be computed from G_r, T, and T_r in $O(n)$ time.

   By Lemma 4, it takes $O(1)$ time to determine whether v_i is special by fetching $S'_6[i]$ from $Z(S'_6)$. If $S'_6[i] = 1$, then $port_r(v_i) = S''_6\langle rank(S'_6, i)\rangle$, which by Lemma 4 can be obtained from S_6 in $\tilde{O}(1)$ time.
7. Let $S_7 = S'_7 \circ S''_7$, where S'_7 and S''_7 are the binary strings such that
   - for each $j_1 = 0, 1, 2, \ldots, \lfloor \frac{n}{L} \rfloor$, $S'_7\langle j_1 \rangle$ keeps the number of r's neighbors in T_r from v_1 to $v_{j_1 \cdot L}$; and
   - for each $j_2 = 0, 1, 2, \ldots, \lfloor \frac{n}{\ell} \rfloor$, $S''_7\{j_2\}$ keeps the number of r's neighbors in T_r from $v_{L\cdot\lfloor j_2\cdot\ell/L\rfloor+1}$ to $v_{j_2\cdot\ell}$.

   Observe that S_7 can be computed from G_r, T, and T_r in $O(n)$ time and $|S_7| = o(n)$.

   For each neighbor v_i of r in T_r, it takes $\tilde{O}(1)$ time to determine $port_r(v_i)$ from S_1 and S_7 as follows. Let $j_1 = \lfloor \frac{i}{L} \rfloor$ and $j_2 = \lfloor \frac{i}{\ell} \rfloor$. For each index $j = j_2\cdot\ell+1, j_2\cdot\ell+2, \ldots, i$, we determine in $O(1)$ time from S_1 whether v_j is a neighbor of r in T_r. As a result, we obtain the number j_3 of neighbors of r in T_r from $v_{j_2\cdot\ell+1}$ to v_i in $\tilde{O}(1)$ time. We have $port_r(v_i) = S'_7\langle j_1\rangle + S''_7\{j_2\} + j_3$.

LEMMA 6. *Our design of routing tables satisfies the following statements.*
1. *The label of each node takes $\lceil \log n \rceil$ bits.*
2. *For each node r, the routing table R_r is computable in $O(n)$ time.*
3. *For each node r, we have $|R_r| < 7.181n + o(n)$.*
4. *The overall bit count of all n routing tables satisfies $\sum_{r\in V} |R_r| \le 7n^2 + o(n^2)$.*

Proof. Statements 1 and 2 are immediate from the construction of R_r.

As for statement 3, by $|S_4| + |S_5| + |S_6| + |S_7| = o(n)$, we have

$$
\begin{aligned}
|R_r| &= |S_1| + |S_2| + |S_3| + o(n) \\
&\le 3m_r + n + \min\left(n, (2n - m_r)\log\frac{en}{2n - m_r}\right) + o(n) \qquad (1)\\
&\le 3m_r + n + (2n - m_r)\log\frac{en}{2n - m_r} + o(n) \\
&= 7n + t\cdot\left(\log\frac{en}{t} - 3\right) + o(n),
\end{aligned}
$$

where $t = 2n - m_r$. By verifying that the maximum of $t\cdot(\log\frac{en}{t} - 3)$ occurs at $t = \frac{1}{8}n$, one can see $|R_r| \le (7 + \frac{1}{8}\log e)n + o(n) < 7.181n + o(n)$.

As for statement 4, by (1), we also have $|R_r| \le 3m_r + 2n + o(n)$, which implies

$$\sum_{r\in V} |R_r| \le 2n^2 + o\left(n^2\right) + 3\sum_{r\in V} m_r. \qquad (2)$$

Note that the above analysis holds for each $T \in \{T^a, T^b, T^c\}$. For each $x \in \{a, b, c\}$, let m_r^x be the number of edges in $T_r \cup T^x$. Thus, $T_r \cap T^x$ has $2(n-1) - m_r^x$ edges. Since each edge of T_r appears in at least one of T^a, T^b, and T^c, we have $6n-6-(m_r^a+m_r^b+m_r^c) \ge n - 1$, which implies

$$\sum_{r\in V} \left(m_r^a + m_r^b + m_r^c\right) \le 5n^2.$$

Therefore, there is an orderly spanning tree $T \in \{T^a, T^b, T^c\}$ of G satisfying

$$\sum_{r\in V} m_r \le \frac{5n^2}{3}. \qquad (3)$$

Combining (2) and (3), we have $\sum_{r\in V} |R_r| \le 7n^2 + o(n^2)$. ∎

4. Determining the correct port efficiently.

LEMMA 7. *It takes $\tilde{O}(\log n)$ time to determine $port_r(v_i)$ from R_r and i.*

Proof. To determine $dad_r(v_k)$ from R_r and k with $v_k \ne r$ in $\tilde{O}(1)$ time, we first determine whether v_k is popular from S_4 in $O(1)$ time.

- If v_k is popular, then we obtain $dad_r(v_k)$ from S_4 in $\tilde{O}(1)$ time.
- If v_k is not popular, we then determine whether v_k is lonely from S_5 in $O(1)$ time.
  - If v_k is not lonely, then we obtain $dad_r(v_k)$ from S_1 and S_5 in $\tilde{O}(1)$ time.
  - If v_k is lonely, then the degree of v_k in G is $\tilde{O}(1)$. We can afford to check the direction of each incident edge of v_k in G. Specifically, for each index $j \ge 2$, it takes $O(1)$ time to obtain $v_b = nbr(v_k, j)$ from S_1 in $O(1)$ time. The direction of (v_k, v_b) can then be determined from S_1, S_2, and S_3 in $O(1)$ time:
    - ∗ If $v_b \in U_<(v_k)$, we compute $q = id(v_k, v_b)$ from S_1 in $O(1)$ time. If $S_2[q] = 1$, then edge (v_b, v_k) is forward, thereby $dad_r(v_k) = v_b$.
    - ∗ If $v_b \in U_>(v_k)$, we compute $q = id(v_k, v_b)$ from S_1 in $O(1)$ time. If $S_2[q] = 0$, then edge (v_k, v_b) is backward, thereby $dad_r(v_k) = v_b$.

∗ If $v_b \in C(v_k)$, we fetch $S'_3[b]$ from S_3 in $O(1)$ time. If $S'_3[b] = 1$, we know that edge (v_k, v_b) is a backward edge of T_r, thereby $dad_r(v_k) = v_b$.

If $nbr(v_k, j) \neq dad_r(v_k)$ for all indices $j \geq 2$, then we have $dad_r(v_k) = nbr(v_k, 1) = mom(v_k)$, which by Lemma 5 can be obtained from S_1 in $O(1)$ time. Therefore, it takes $\tilde{O}(1)$ time to obtain $dad_r(v_k)$ from S_1, S_2, and S_3.

It takes $O(1)$ time to determine whether v_k is a neighbor of r in T_r from S_1 and k. It also takes $O(1)$ time to determine whether v_k is special from S_6 and k. Since $dad_r(v_k)$ is obtainable from R_r and k, with $v_k \neq r$, in $\tilde{O}(1)$ time, it follows from the choice of special nodes that we can traverse T_r from v_i toward r reaching in $\tilde{O}(L) = \tilde{O}(\log n)$ time a node v_j that is either a special node or a neighbor of r in T_r. Note that $port_r(v_i) = port_r(v_j)$. If v_j is a special node, then $port_r(v_j)$ can be obtained from S_6 in $\tilde{O}(1)$ time. If v_j is a neighbor of r in T_r, then $port_r(v_j)$ can be obtained from S_7 in $\tilde{O}(1)$ time. The lemma is proved. □

The following main theorem of the paper is immediate from Lemmas 6 and 7.

THEOREM 8. *Given an n-node planar network G and a routing spanning tree T_r for each node r of G, there is a design of routing tables such that*

- *the label of each node takes $\lceil \log n \rceil$ bits;*
- *the routing table of each node can be encoded in at most $7.181n + o(n)$ bits;*
- *the routing tables of all nodes can be encoded in at most $7n^2 + o(n^2)$ bits;*
- *the time to obtain each routing table for node r from G and T_r is $O(n)$; and*
- *the time to obtain the port index from a routing table and the label of a packet destination is $\tilde{O}(\log n)$;*

where the time complexity is measured under the $O(\log n)$-bit unit-cost RAM model.

5. Concluding remarks.

Some questions might be worth further investigation. For example, closing the gap between the lower and upper bounds on $\rho_r(n)$ is an interesting one. Also, if $\lambda_r(n) > \lceil \log n \rceil$; e.g., $\lambda_r(n) = 3\lceil \log n \rceil$ or $\lambda_r(n) = O((\log n)^{O(1)})$, can one significantly improve the upper and lower bounds on $\rho_r(n)$?

Acknowledgments. We thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] I. Abraham, C. Gavoille, and D. Malkhi, *Compact routing for graphs excluding a fixed minor*, in Proceedings of the 19th International Conference on Distributed Computing, Lecture Notes in Comput. Sci. 3724, Springer-Verlag, New York, 2005, pp. 442–456.

[2] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg, *Improved routing strategies with succinct tables*, J. Algorithms, 11 (1990), pp. 307–341.

[3] M. Becker and K. Mehlhorn, *Algorithms for routing in planar graphs*, Acta Inform., 23 (1986), pp. 163–176.

[4] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression*, Prentice-Hall, Englewood Cliffs, NJ, 1990.

[5] P. Bose and P. Morin, *Online routing in triangulations*, in Proceedings of the 10th International Symposium on Algorithms and Computation, Lecture Notes in Comput. Sci. 1741, Springer-Verlag, 1999, New York, pp. 113–122.

[6] P. Bose and P. Morin, *Competitive online routing in geometric graphs*, in Proceedings of the 8th International Colloquium on Structural Information and Communication Complexity, 2001, pp. 35–44.

[7] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, *Routing with guaranteed delivery in ad hoc wireless networks*, Wireless Netw., 7 (2001), pp. 609–616.

[8] L. P. Chew, *There are planar graphs almost as good as the complete graph*, J. Comput. System Sci., 39 (1989), pp. 205–219.

[9] Y.-T. Chiang, C.-C. Lin, and H.-I. Lu, *Orderly spanning trees with applications to graph drawing and graph encoding*, in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, Washington, DC, 2001, pp. 506–515.

[10] Y.-T. Chiang, C.-C. Lin, and H.-I. Lu, *Orderly spanning trees with applications*, SIAM J. Comput., 34 (2005), pp. 924–945.

[11] M. Chrobak and T. H. Payne, *A linear-time algorithm for drawing a planar graph on a grid*, Inform. Process. Lett., 54 (1995), pp. 241–246.

[12] R. C.-N. Chuang, A. Garg, X. He, M.-Y. Kao, and H.-I. Lu, *Compact encodings of planar graphs via canonical ordering and multiple parentheses*, in Proceedings of the 25th International Colloquium on Automata, Languages, and Programming, K. G. Larsen, S. Skyum, and G. Winskel, eds., Lecture Notes in Comput. Sci. 1443, Springer-Verlag, New York, 1989, pp. 118–129.

[13] L. Cowen and C. G. Wagner, *Compact roundtrip routing in directed networks*, J. Algorithms, 50 (2004), pp. 79–95.

[14] L. J. Cowen, *Compact routing with minimum stretch*, J. Algorithms, 38 (2001), pp. 170–183.

[15] H. de Fraysseix, J. Pach, and R. Pollack, *How to draw a planar graph on a grid*, Combinatorica, 10 (1990), pp. 41–51.

[16] Y. Dourisboure and C. Gavoille, *Improved compact routing scheme for chordal graphs*, in Proceedings of the 16th International Conference on Distributed Computing, D. Malkhi, ed., Lecture Notes in Comput. Sci. 2508, Springer-Verlag, New York, 2002, pp. 252–264.

[17] T. Eilam, C. Gavoille, and D. Peleg, *Compact routing schemes with low stretch factor*, J. Algorithms, 46 (2003), pp. 97–114.

[18] T. Eilam, C. Gavoille, and D. Peleg, *Average stretch analysis of compact routing schemes*, Discrete Appl. Math., 155 (2007), pp. 598–610.

[19] P. Elias, *Universal codeword sets and representations of the integers*, IEEE Trans. Inform. Theory, IT-21 (1975), pp. 194–203.

[20] P. Fraigniaud and C. Gavoille, *Routing in trees*, in Proceedings of the 28th International Colloquium on Automata, Languages and Programming, F. Orejas, P. G. Spirakis, and J. van Leeuwen, eds., Lecture Notes in Comput. Sci. 2076, Springer-Verlag, New York, pp. 757–772.

[21] P. Fraigniaud and C. Gavoille, *A space lower bound for routing in trees*, in Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Comput. Sci. 2285, Springer-Verlag, New York, pp. 65–75.

[22] G. N. Frederickson and R. Janardan, *Designing networks with compact routing tables*, Algorithmica, 3 (1988), pp. 171–190.

[23] G. N. Frederickson and R. Janardan, *Efficient message routing in planar networks*, SIAM J. Comput., 18 (1989), pp. 843–857.

[24] M. L. Fredman and D. E. Willard, *Trans-dichotomous algorithms for minimum spanning trees and shortest paths*, J. Comput. System Sci., 48 (1994), pp. 533–551.

[25] J. Gao, L. J. Guibas, J. Hershburger, L. Zhang, and A. Zhu, *Geometric spanners for routing in mobile networks*, IEEE J. Selected Areas Communications, 23 (2005), pp. 174–185.

[26] C. Gavoille, *A survey on interval routing*, Theoret. Comput. Sci., 245 (2000), pp. 217–253.

[27] C. Gavoille and M. Gengler, *Space-efficiency of routing schemes of stretch factor three*, J. Parallel Distrib. Comput., 61 (2001), pp. 679–687.

[28] C. Gavoille and N. Hanusse, *Compact routing tables for graphs of bounded genus*, in 26th International Colloquium on Automata, Languages and Programming, J. Wiedermann, P. van Emde Boas, and M. Nielsen, eds., Lecture Notes in Comput. Sci. 1644, Springer-Verlag, New York, pp. 351–360.

[29] C. Gavoille and D. Peleg, *The compactness of interval routing*, SIAM J. Discrete Math., 12 (1999), pp. 459–473.

[30] C. Gavoille and D. Peleg, *The compactness of interval routing for almost all graphs*, SIAM J. Comput., 31 (2001), pp. 706–721.

[31] R. F. Geary, N. Rahman, R. Raman, and V. Raman, *A simple optimal representation for balanced parentheses*, Theoret. Comput. Sci. 368 (2006), pp. 231–246.

[32] T. Hagerup, P. B. Miltersen, and R. Pagh, *Deterministic dictionaries*, J. Algorithms, 41 (2001), pp. 69–85.

[33] D. Harel and M. Sardas, *An algorithm for straight-line drawing of planar graphs*, Algorithmica, 20 (1998), pp. 119–135.

[34] Y. Hassin and D. Peleg, *Sparse communication networks and efficient routing in the plane*, Distrib. Comput., 14 (2001), pp. 205–215.

[35] M. R. Henzinger, P. N. Klein, S. Rao, and S. Subramanian, *Faster shortest-path algorithms for planar graphs*, J. Comput. Syst. Sci., 55 (1997), pp. 3–23.

[36] G. Jacobson, *Space-efficient static trees and graphs*, in Proceedings of the 30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, NC, 1989, IEEE, pp. 549–554.

[37] G. Kant, *Drawing planar graphs using the canonical ordering*, Algorithmica, 16 (1996), pp. 4–32.

[38] G. Kant and X. He, *Regular edge labeling of* 4*-connected plane graphs and its applications in graph drawing problems*, Theoret. Comput. Sci., 172 (1997), pp. 175–193.

[39] B. N. Karp, *Geographic Routing for Wireless Networks*, Ph.D. thesis, Harvard University, Cambridge, MA, 2000.

[40] B. N. Karp and H. T. Kung, *GPSR: Greedy perimeter stateless routing for wireless networks*, in Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), Boston, 2000, pp. 243–254.

[41] D. Krioukov, K. C. Claffy, K. Fall, and A. Brady, *On compact routing for the internet*, ACM SIGCOMM Comput. Commun. Rev., 37 (2007), pp. 41–52.

[42] D. V. Krioukov, K. R. Fall, and X. Yang, *Compact routing on internet-like graphs*, in Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom), 2004, pp. 209–218.

[43] X.-Y. Li, G. Calinescu, and P.-J. Wan, *Distributed construction of a planar spanner and routing for ad hoc wireless networks*, in Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom), Vol. 3, New York, 2002, pp. 1268–1277.

[44] C.-C. Liao, H.-I. Lu, and H.-C. Yen, *Floor-planning via orderly spanning trees*, in Proceedings of the 9th International Symposium on Graph Drawing, Lecture Notes in Comput. Sci. 2265, Springer-Verlag, New York, 2001, pp. 367–377.

[45] C.-C. Liao, H.-I. Lu, and H.-C. Yen, *Compact floor-planning via orderly spanning trees*, J. Algorithms, 48 (2003), pp. 441–451.

[46] G. Lin, *Fault tolerant planar communication networks*, in Proceedings of the 24th Annual ACM Symposium on the Theory of Computing, 1992, pp. 133–139.

[47] H.-I. Lu, *Improved compact routing tables for planar networks via orderly spanning trees*, in Proceedings of the 8th International Conference on Computing and Combinatorics, O. H. Ibarra and L. Zhang, eds., Lecture Notes in Computer Science 2387, Springer-Verlag, London, pp. 57–66.

[48] H.-I. Lu and C.-C. Yeh, *Balanced parentheses strike back*, ACM Trans. Algorithms, 4 (2008), pp. 28.1–28.13.

[49] J. S. B. Mitchell, *Geometric shortest paths and network optimization*, in Handbook of Computational Geometry, J.-R. Sack and J. Urrutia, eds., Elsevier Science, Amsterdam, 2000, pp. 633–702.

[50] P. R. Morin, *Online Routing in Geometric Graphs*, Ph.D. thesis, Carleton University, Ottawa, ON, 2001.

[51] J. I. Munro and V. Raman, *Succinct representation of balanced parentheses and static trees*, SIAM J. Comput., 31 (2001), pp. 762–776.

[52] C.-I. Nakano, *Planar drawings of plane graphs*, IEICE Trans. Inform. Systems, E83-D (2000), pp. 384–391.

[53] G. Narasimhan and M. Smid, *Approximating the stretch factor of Euclidean graphs*, SIAM J. Comput., 30 (2000), pp. 978–989.

[54] R. Pagh, *Low redundancy in static dictionaries with constant query time*, SIAM J. Comput., 31 (2001), pp. 353–363.

[55] D. Peleg, *Distributed computing: A locality-sensitive approach*, Vol. 5, SIAM Monogr. Discrete Math. Appl., SIAM, Philadelphia, 2000.

[56] D. Peleg and E. Upfal, *A trade-off between space and efficiency for routing tables*, J. ACM, 36 (1989), pp. 510–530.

[57] W. Schnyder, *Planar graphs and poset dimension*, Order, 5 (1989), pp. 323–343.

[58] W. Schnyder, *Embedding planar graphs on the grid*, in Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, 1990, pp. 138–148.

[59] L. Subramanian, V. N. Padmanabhan, and R. H. Katz, *Geographic properties of internet routing*, in Proceedings of the USENIX Annual Technical Conference, Berkeley, CA, 2002, USENIX Association, pp. 243–259.

[60] M. Thorup, *Compact oracles for reachability and approximate distances in planar digraphs*, J. ACM, 51 (2004), pp. 993–1024.
[61] M. Thorup, *Compact oracles for reachability and approximate distances in planar digraphs*, J. ACM, 51 (2004), pp. 993–1024.
[62] M. Thorup and U. Zwick, *Compact routing schemes*, in Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures, 2001, ACM Press, pp. 1–10.
[63] W. T. Tutte, *A census of planar triangulations*, Canad. J. Math., 14 (1962), pp. 21–38.
[64] P. van Emde Boas, *Machine models and simulations*, in Handbook of Theoretical Computer Science, Vol. A, J. van Leeuwen, ed., Elsevier, Amsterdam, 1990, pp. 1–60.
[65] M. Yannakakis, *Embedding planar graphs in four pages*, J. Comput. System Sci., 38 (1989), pp. 36–67.