SIAM J. COMPUT.
Vol. 43, No. 2, pp. 477–496

LINEAR-TIME COMPRESSION OF BOUNDED-GENUS GRAPHS INTO INFORMATION-THEORETICALLY OPTIMAL NUMBER OF BITS*

HSUEH-I LU†

Abstract. A *compression scheme* A for a class $\mathbb{G}$ of graphs consists of an encoding algorithm $Encode_A$ that computes a binary string $Code_A(G)$ for any given graph G in $\mathbb{G}$ and a decoding algorithm $Decode_A$ that recovers G from $Code_A(G)$. A compression scheme A for $\mathbb{G}$ is *optimal* if both $Encode_A$ and $Decode_A$ run in linear time and the number of bits of $Code_A(G)$ for any n-node graph G in $\mathbb{G}$ is information-theoretically optimal to within lower-order terms. Trees and plane triangulations were the only known nontrivial graph classes to admit optimal compression schemes. Based upon Goodrich's separator decomposition for planar graphs and Djidjev and Venkatesan's planarizers for bounded-genus graphs, we give an optimal compression scheme for any hereditary (i.e., closed under taking subgraphs) class $\mathbb{G}$ under the premise that any n-node graph of $\mathbb{G}$ to be encoded comes with a genus-$o(\frac{n}{\log^2 n})$ embedding. By Mohar's linear-time algorithm that embeds a bounded-genus graph on a genus-$O(1)$ surface, our result implies that any hereditary class of genus-$O(1)$ graphs admits an optimal compression scheme. For instance, our result yields the first-known optimal compression schemes for planar graphs, plane graphs, graphs embedded on genus-1 surfaces, graphs with genus 2 or less, 3-colorable directed plane graphs, 4-outerplanar graphs, and forests with degree at most 5. For nonhereditary graph classes, we also give a methodology for obtaining optimal compression schemes. From this methodology, we give the first-known optimal compression schemes for triangulations of genus-$O(1)$ surfaces and floorplans.

Key words. trees, planar graphs, graph algorithms, data structures, compression

AMS subject classifications. 05C05, 05C10, 05C85, 68P05, 68P30

DOI. 10.1137/120879142

1. Introduction.

Compact representations of graphs are fundamentally important and useful in many applications, including representing the meshes in finite element analysis, terrain models of GIS, three-dimensional (3D) models of graphics [48, 64, 80, 81, 82, 85, 89, 92], and VLSI design [56, 84], designing compact routing tables of computer networks [1, 3, 16, 35, 36, 38, 66, 77, 94, 95], and compressing the link structure of the Internet [2, 5, 7, 15, 21, 88]. Let $\mathbb{G}$ be a class of graphs. Let $num(\mathbb{G}, n)$ denote the number of distinct n-node graphs in $\mathbb{G}$. The information-theoretically optimal number of bits to encode an n-node graph in $\mathbb{G}$ is $\lceil \log num(\mathbb{G}, n)\rceil$.[1] For instance, if $\mathbb{G}$ is the class of rooted trees, then $num(\mathbb{G}, n) \approx \frac{2^{2n}}{n^{3/2}}$ and $\log num(\mathbb{G}, n) = 2n - O(\log n)$; if $\mathbb{G}$ is the class of plane triangulations, then $\log num(\mathbb{G}, n) = \log \frac{256}{27} n + o(n) \approx 3.2451n + o(n)$ [97]. A *compression scheme* A for $\mathbb{G}$ consists of an encoding algorithm $Encode_A$ that computes a binary string $Code_A(G)$ for any given graph G in $\mathbb{G}$ and a decoding algorithm $Decode_A$ that recovers graph G from $Code_A(G)$. A compression scheme A for a graph class $\mathbb{G}$ with $\log num(\mathbb{G}, n) = O(n)$

*Received by the editors May 29, 2012; accepted for publication (in revised form) January 9, 2014; published electronically March 20, 2014. A preliminary version of this paper appeared in *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2002, pp. 223–224. This research was supported in part by NSC grant 101–2221–E–002–062–MY3.

http://www.siam.org/journals/sicomp/43-2/87914.html

†Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan, ROC (hil@csie.ntu.edu.tw, http://www.csie.ntu.edu.tw/∼hil/). The author also holds joint appointments from the Graduate Institute of Networking and Multimedia and the Graduate Institute of Biomedical Electronics and Bioinformatics, National Taiwan University.

[1] All logarithms throughout the paper are to the base of two.

is *optimal* if the following three conditions hold:

Condition C1. The running time of algorithm $\mathit{Encode}_A(G)$ is linear in the size of G.

Condition C2. The running time of algorithm $\mathit{Decode}_A(\mathit{Code}_A(G))$ is linear in the bit count of $\mathit{Code}_A(G)$.

Condition C3. For all positive constants β with $\log \mathit{num}(\mathbb{G}, n) \leq \beta n + o(n)$, the bit count of $\mathit{Code}_A(G)$ for an n-node graph G in $\mathbb{G}$ is no more than $\beta n + o(n)$.

Note that Condition C3 basically says the bit count of $\mathit{Code}_A(G)$ is information-theoretically optimal to within lower-order terms. Although there has been considerable work on compression schemes, trees (see, e.g., [11, 50, 67, 72]) and plane triangulations [79] were the only known nontrivial graph classes to admit optimal compression schemes. A graph class is *hereditary* if it is closed under taking subgraphs. Below is the main result of the paper.

THEOREM 1.1. *Any hereditary class $\mathbb{G}$ of graphs with $\log \mathit{num}(\mathbb{G}, n) = O(n)$ admits an optimal compression scheme, as long as each input n-node graph in $\mathbb{G}$ to be encoded comes with a genus-$o(\frac{n}{\log^2 n})$ embedding.*

By Theorem 1.1 and Mohar's linear-time genus-$O(1)$ embedding algorithm for genus-$O(1)$ graphs [54, 70] (see Lemma 2.5), any hereditary class of genus-$O(1)$ graphs admits an optimal compression scheme. For instance, our result yields the first-known optimal compression schemes for planar graphs, plane graphs, graphs embedded on genus-1 surfaces, graphs with genus 2 or less, 3-colorable directed plane graphs, 4-outerplanar graphs, and forests with degree at most 5. For nonhereditary graph classes, we also give an extension (see Corollary 5.1) of Theorem 1.1. As summarized in the following theorem, we show two classes of genus-$O(1)$ graphs whose optimal compression schemes are obtainable via this extension, where the class of floorplans is defined in related work below.

THEOREM 1.2. *The following two classes of graphs admit optimal compression schemes:*

(1) *triangulations of a genus-g surface for any integral constant g,*

(2) *floorplans.*

Technical overview. The kernel of the proof of Theorem 1.1 is a linear-time disjoint partition $G_0, \ldots, G_p$ of an n-node graph G embedded on a genus-$o(\frac{n}{\log^2 n})$ surface.[2] Let poly(n) denote $O(n^{O(1)})$. Based upon Goodrich's separator decomposition of planar graphs [40] and Djidjev and Venkatesan's planarizer [26], partition $G_0, \ldots, G_p$ satisfies the following conditions, where n_i is the number of nodes of G_i and d_i is the number of times that the nodes of G_i are duplicated in some G_j with $j \neq i$:[3] (a) $n_0 = o(\frac{n}{\log n})$, (b) $n_i = \text{poly}(\log n)$ holds for each $i = 1, 2, \ldots, p$, (c) $\sum_{i=1}^{p} d_i = o(\frac{n}{\log n})$, and (d) $\sum_{i=0}^{p} n_i = n + o(\frac{n}{\log n})$. By condition (a), G_0 can be encoded in $o(n)$ bits. By conditions (b) and (c), the information required to recover G from $G_0, G_1, \ldots, G_p$ can be encoded into $o(n)$ bits (see Lemma 4.1). By condition (d), we have $\log \mathit{num}(\mathbb{G}, n) \leq o(n) + \sum_{i=1}^{p} \log \mathit{num}(\mathbb{G}, n_i)$. Therefore, the disjoint partition reduces the problem of encoding an n-node graph in $\mathbb{G}$ to the problem of encoding a poly$(\log n)$-node graph in $\mathbb{G}$. Applying such a reduction for one more level, it remains to encode a poly$(\log \log n)$-node graph in $\mathbb{G}$ into an information-theoretically optimal

[2]Precisely, the disjoint partition $G_0, \ldots, G_p$ of the edges of the embedded graph G in the proof of Theorem 1.1 is $G[V_0], G(V_1), \ldots, G(V_p)$, where $[V_0, \ldots, V_p]$ is both (i) a 1-separation $\mathbb{S}_1$ of an arbitrary triangulation Δ of G and (ii) a refinement of the 0-separation $\mathbb{S}_0 = [\varnothing, \mathit{Node}(\Delta)]$ of Δ.

[3]As a matter of fact, in our construction, all duplicated nodes of G_i with $i \geq 1$ belong to G_0.

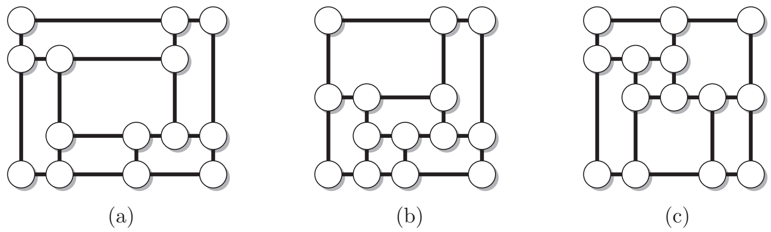


FIG. 1. *Three floorplans with* 14 *nodes,* 6 *internal faces, and* 19 *edges. Floorplans* (a) *and* (b) *are equivalent, and floorplans* (b) *and* (c) *are not equivalent.*

number of bits, which can be resolved by the standard technique (see, e.g., [47, 72, 78]) of precomputation tables (see Lemma 2.3).

Related work. The compression scheme of Turán [96] encodes an n-node plane graph that may have self-loops into $12n$ bits.[4] Keeler and Westbrook [55] improved this bit count to $10.74n$. They also gave compression schemes for several families of plane graphs. In particular, they used $4.62n$ bits for plane triangulation, and $9n$ bits for connected plane graphs free of self-loops and degree-one nodes. For plane triangulations, He, Kao, and Lu [46] improved the bit count to $4n$. For triconnected plane graphs, He, Kao, and Lu [46] also improved the bit count to at most $8.585n$ bits. This bit count was later reduced to at most $\frac{9\log_2 3}{2}n \approx 7.134n$ by Chuang et al. [20]. For any given n-node graph G embedded on a genus-g surface, Deo and Litow [25] showed an $O(ng)$-bit encoding for G. These compression schemes all take linear time for encoding and decoding, but Condition C3 does not hold for them. The compression schemes of He, Kao, and Lu [47] (respectively, Blelloch and Farzan [14]) for planar graphs, plane graphs, and plane triangulations (respectively, separable graphs) satisfy Condition C3, but their encoding algorithms require $\Omega(n \log n)$ time on n-node graphs.

Floorplanning is a fundamental issue in circuit layout [4, 8, 17, 24, 32, 43, 51, 57, 58, 62, 68, 69, 84, 91, 106, 108]. Motivated by VLSI physical design, various representations of floorplans were proposed [33, 109, 110]. Designing a floorplan to meet a certain criterion is NP-complete in general [44, 87, 100], so heuristic techniques such as simulated annealing [17, 101, 102] are practically useful. The length of the encoding affects the size of the search space. A *floorplan*, which is also known as *rectangular drawing*, is a division of a rectangle into rectangular faces using horizontal and vertical line segments. Two floorplans are *equivalent* if they have the same adjacency relations and relative positions among the nodes. For instance, Figure 1 shows three floorplans: Floorplans (a) and (b) are equivalent. Floorplans (b) and (c) are not equivalent. Let G be the input n-node floorplan. Under the conventional assumption that each node of G, other than the four corner nodes, has exactly three neighbors (see, e.g., [45, 107]), one can verify that G has $0.5n$ faces and $1.5n-2$ edges. Yamanaka and Nakano [103] showed how to encode G into $2.5n$ bits. Chuang [19] reduced the bit count to $2.293n$. Takahashi, Fujimaki, and Inoue [90] further reduced the bit count to $2n$. All these compression schemes for floorplans satisfy Conditions C1 and C2, but not Condition C3. Takahashi, Fujimaki, and Inoue [90] also showed that the number of distinct n-node floorplans is no more than $3.375^{n+o(n)} \approx 2^{1.755n+o(n)}$. Therefore, our Theorem 1.2(2) encodes an n-node floorplan into at most $1.755n$ bits.

For applications that require query support, Jacobson [50] gave a $\Theta(n)$-bit en-

[4] For brevity, we omit all lower-order terms of bit counts in our discussion of related work.

coding for a connected and simple planar graph G that supports traversal in $\Theta(\log n)$ time per node visited. Munro and Raman [71] improved this result and gave schemes to encode binary trees, rooted ordered trees, and planar graphs. For a general n-node m-edge planar graph G, they used $2m + 8n$ bits while supporting adjacency and degree queries in $O(1)$ time. Chuang et al. [20] reduced this bit count to $2m + (5 + \frac{1}{k})n$ for any constant $k > 0$ with the same query support. The bit count can be further reduced if only $O(1)$-time adjacency queries are supported, or if G is simple, triconnected, or triangulated [20]. Chiang, Lin, and Lu [18] reduced the number of bits to $2m + 2n$. Yamanaka and Nakano [105] showed a $6n$-bit encoding for plane triangulations with query support. The succinct encodings of Blandford, Blelloch, and Kash [13] and Blelloch and Farzan [14] for separable graphs support queries. Yamanaka and Nakano [104] also gave a compression scheme for floorplans with query support. For labeled planar graphs, Itai and Rodeh [49] gave an encoding of $\frac{3}{2}n \log n$ bits. For unlabeled general graphs, Naor [74] gave an encoding of $\frac{1}{2}n^2$ bits. For certain graph families, Kannan, Naor, and Rudich [52] gave schemes that encode each node with $O(\log n)$ bits and support $O(\log n)$-time testing of adjacency between two nodes. Galperin and Wigderson [34] and Papadimitriou and Yannakakis [75] investigated complexity issues arising from encoding a graph by a small circuit that computes its adjacency matrix. Related work on various versions of succinct graph representations can be found in [6, 9, 28, 29, 30, 31, 37, 42, 53, 73, 76, 83] and the references therein.

Outline. The rest of the paper is organized as follows. Section 2 gives the preliminaries. Section 3 shows our algorithm for computing graph separations. Section 4 gives our optimal compression scheme for hereditary graph classes. Section 5 shows a methodology for obtaining optimal compression schemes for nonhereditary graph classes and applies this methodology on triangulations of genus-$O(1)$ graphs and floorplans. Section 6 concludes the paper with a couple of open questions.

2. Preliminaries.

Unless clearly stated otherwise, all graphs throughout the paper are simple, i.e., have no multiple edges or self-loops.

2.1. Segmentation prefix.

Let $\|X\|$ denote the number of bits of binary string X. A binary string X_0 is a *segmentation prefix* of binary strings $X_1, \ldots, X_d$ if (a) it takes $O(\sum_{i=1}^{d} \|X_i\|)$ time to compute X_0 from $X_1, \ldots, X_d$ and (b) given the concatenation of $X_0, X_1, \ldots, X_d$, it takes $O(\sum_{i=0}^{d} \|X_i\|)$ time to recover all X_i with $1 \le i \le d$.

LEMMA 2.1 (see, e.g., [10, 27]). *Any binary strings $X_1, \ldots, X_d$ with $d = O(1)$ have a segmentation prefix with $O(\log \sum_{i=1}^{d} \|X_i\|)$ bits.*

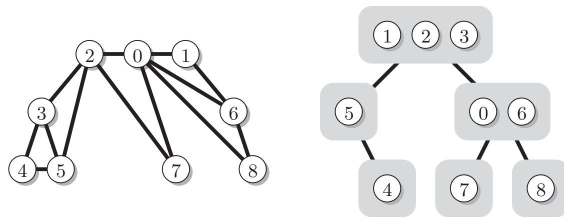
LEMMA 2.2. *Any binary strings $X_1, X_2, \ldots, X_d$ have an $O(\min\{m,\ d \log m\})$-bit segmentation prefix, where $m = \|X_1\| + \cdots + \|X_d\|$.*

Proof. Let X be the concatenation of $X_1, \ldots, X_d$. If $m \le d \log m$, let X' be the m-bit binary string with exactly d copies of 1-bits such that the jth bit of X' is 1 if and only if $j = \|X_1\| + \cdots + \|X_i\|$ holds for some $i = 1, \ldots, d$. Otherwise, let X' store the $O(\log m)$-bit numbers $\|X_1\| + \cdots + \|X_i\|$ for all $i = 1, \ldots, d$. Let X'_0 be the segmentation prefix of X' and X as ensured by Lemma 2.1. The concatenation of X'_0 and X' is a segmentation prefix X_0 of $X_1, \ldots, X_d$ with $O(\min\{m,\ d \log m\})$ bits. The lemma is proved. ∎

For the rest of the paper, let $X_1 \circ \cdots \circ X_d$ be the concatenation of $X_0, X_1, \ldots, X_d$, where X_0 is the segmentation prefix of $X_1, \ldots, X_d$ as ensured by Lemma 2.2.

2.2. Precomputation table.

Let $|S|$ denote the cardinality of set S. Let $\mathit{Node}(G)$ consist of the nodes in graph G, and let $\mathit{node}(G) = |\mathit{Node}(G)|$. For any subset V of $\mathit{Node}(G)$, let $G[V]$ denote the subgraph of G induced by V, and let $G \setminus V$



(a) (b)

FIG. 2. (a) *A* 9*-node plane graph* G. (b) *A separator decomposition* $\mathbb{T}$ *of* G.

denote the subgraph of G obtained by deleting V and their incident edges. Two disjoint subsets V and V' of $Node(G)$ are *adjacent* in G if there is an edge (v, v') of G with $v \in V$ and $v' \in V'$. For any subset V of $Node(G)$, let $Nbr_G(V)$ consist of the nodes in $Node(G) \setminus V$ that are adjacent to V in G, and let $nbr_G(V) = |Nbr_G(V)|$. A *connected component* of graph G is a maximal subset C of $Node(G)$ such that $G[C]$ is connected.

LEMMA 2.3. *Let* $\mathbb{G}$ *be a graph class satisfying* $\log num(\mathbb{G}, n) = O(n)$. *Given positive integers* ℓ *and* n *with* $\ell = poly(\log\log n)$, *it takes overall* $o(n)$ *time to compute* (i) *a labeling* $Label(H)$ *and a* $\lceil \log num(\mathbb{G}, node(H)) \rceil$*-bit binary string* $Optcode(H)$ *for each distinct graph* $H \in \mathbb{G}$ *with at most* ℓ *nodes and* (ii) *an* $o(n)$*-bit string* $Table(\mathbb{G}, \ell)$ *such that the following statements hold:*

(1) *Given a graph* $H \in \mathbb{G}$ *with* $node(H) \le \ell$, *it takes* $O(node(H))$ *time to obtain* $Optcode(H)$ *and* $Label(H)$ *from* $Table(\mathbb{G}, \ell)$.
(2) *Given* $Optcode(H)$ *for a graph* $H \in \mathbb{G}$ *with* $node(H) \le \ell$, *it takes* $O(node(H))$ *time to obtain* H *and* $Label(H)$ *from* $Table(\mathbb{G}, \ell)$.

Proof. It is straightforward by $O(1)^{\mathrm{poly}(\ell)} = o(n)$. ∎

2.3. Separator decomposition of planar graphs. Sets $S_1, S_2, \ldots, S_d$ form a *disjoint partition* of set S if $S_1, \ldots, S_d$ are pairwise disjoint and $S = S_1 \cup \cdots \cup S_d$. A subset S of $Node(G)$ is a *separator* of graph G with respect to S_1 and S_2 if (1) S, S_1, and S_2 form a disjoint partition of $Node(G)$, (2) S_1 and S_2 are not adjacent in G, (3) $|S| = O(node(G)^{1/2})$, and (4) $\max\{|S_1|, |S_2|\} \le \frac{2}{3} \cdot node(G)$. A *separator decomposition* [12] of G is a rooted binary tree $\mathbb{T}$ on a disjoint partition of $Node(G)$ such that the following two statements hold, where "nodes" specify elements of $Node(G)$ and "vertices" specify elements of $Node(\mathbb{T})$. Statement 1: Each leaf vertex of $\mathbb{T}$ consists of a single node of G. Statement 2: Each internal vertex S of $\mathbb{T}$ is a separator of $G[Offspring(S)]$ with respect to $Offspring(S_1)$ and $Offspring(S_2)$, where S_1 and S_2 are the child vertices of S in $\mathbb{T}$ and $Offspring(S)$ (respectively, $Offspring(S_1)$ and $Offspring(S_2)$) is the union of all the vertices in the subtree of $\mathbb{T}$ rooted at S (respectively, S_1 and S_2). See Figure 2 for an illustration.

LEMMA 2.4 (Goodrich [40]). *It takes* $O(n)$ *time to compute a separator decomposition for any given* n*-node planar graph.*

2.4. Planarizers for nonplanar graphs. The *genus* of a graph G is defined to be the smallest integer g such that G can be embedded on an orientable surface with g handles without edge crossings [41]. For example, the genus of a planar graph is zero. By Euler's formula (see, e.g., [39]), an n-node genus-$O(n)$ graph has $O(n)$ edges. Determining the genus of a general graph is NP-complete [93], but Mohar [70] showed that it takes linear time to determine whether a graph is of genus g

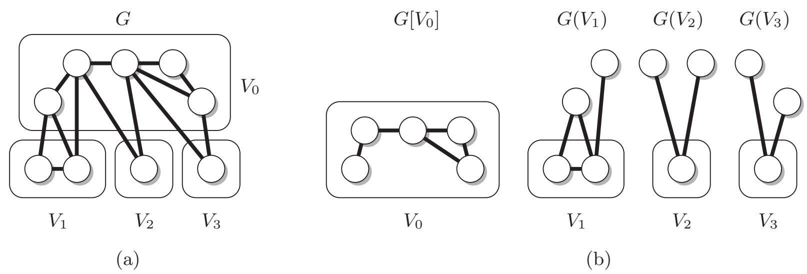


FIG. 3. (a) *A 9-node plane graph with a separation* $[V_0, \ldots, V_3]$. (b) $G[V_0]$*,* $G(V_1)$*,* $G(V_2)$*, and* $G(V_3)$ *form a disjoint partition of the edges of* G*.*

for any $g = O(1)$. Mohar's algorithm is simplified by Kawarabayashi, Mohar, and Reed [54].

LEMMA 2.5 (Kawarabayashi, Mohar, and Reed [54] and Mohar [70]). *It takes* $O(n)$ *time to compute a genus-*$O(1)$ *embedding for any given* n*-node genus-*$O(1)$ *graph.*

Gilbert, Hutchinson, and Tarjan [39] gave an $O(n+g)$-time algorithm to compute an $O((gn)^{0.5})$-node separator of an n-node genus-g graph, generalizing Lipton and Tarjan's classic separator theorem for planar graphs [63]. Our result relies on the following planarization algorithm.

LEMMA 2.6 (Djidjev and Venkatesan [26]). *Given an* n*-node graph* G *embedded on a genus-*g *surface, it takes* $O(n + g)$ *time to compute a subset* V *of* $Node(G)$ *with* $|V| = O((gn)^{0.5})$ *such that* $G \setminus V$ *is planar.*

3. Separation and refinement. We say that $[V_0, V_1, \ldots, V_p]$ with $p \geq 1$ is a *separation* of graph G if the following properties hold:

Property S1. $V_0, V_1, \ldots, V_p$ form a disjoint partition of $Node(G)$.

Property S2. Any two V_i and $V_{i'}$ with $1 \leq i \neq i' \leq p$ are not adjacent in G.

Figure 3(a) shows a separation $[V_0, V_1, V_2, V_3]$ of graph G, and Figure 4(a) shows another separation $[U_0, U_1, U_2]$ of G. For any subset V of $Node(G)$, let $G(V)$ be the subgraph of G induced by $V \cup Nbr_G(V)$ excluding the edges of $G[Nbr_G(V)]$. If $[V_0, \ldots, V_p]$ is a separation of G, then $G[V_0], G(V_1), \ldots, G(V_p)$ form a disjoint partition of the edges of G. See Figures 3(b) and 4(b) for illustrations. Let $\log^{(0)} n = n$. For any positive integer k, let $\log^{(k)} n = \log\,(\log^{(k-1)} n)$. For notational brevity, for any nonnegative integer k, let

$$\ell_k = \max\{1, \log^{(k)} n\}.$$

For any nonnegative integer k, separation $[V_0, \ldots, V_p]$ of an n-node graph G is a k*-separation* of G if the following three properties hold:

Property S3. $|V_0| = o(\frac{n}{\ell_k})$ and $p = o(\frac{n}{\ell_k}) + 1$.

Property S4. $|V_i| + nbr_G(V_i) = \text{poly}(\ell_k)$ holds for each $i = 1, \ldots, p$.

Property S5. $\sum_{i=1}^{p} nbr_G(V_i) = o(\frac{n}{\ell_k})$.

One can easily verify that $[\varnothing, Node(G)]$ is a 0-separation of G.[5] Let $[V_0, \ldots, V_p]$ and $[U_0, \ldots, U_q]$ be two separations of graph G. We say that $[V_0, \ldots, V_p]$ is a *refinement* of $[U_0, \ldots, U_q]$ if the following three properties hold:

[5]The "+1" in Property S3 is redundant for $k \geq 1$. However, we need it so that $[\varnothing, Node(G)]$ is a 0-separation of G, since $1 \neq o(\frac{n}{\ell_0})$.

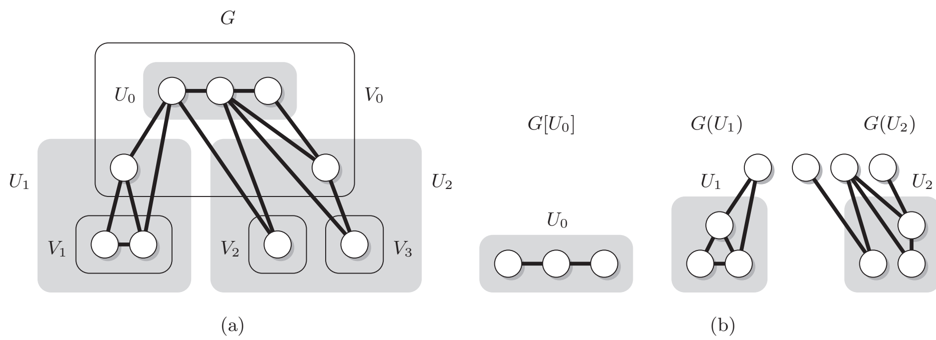


FIG. 4. (a) *Separation* $[V_0, V_1, V_2, V_3]$ *is a refinement of separation* $[U_0, U_1, U_2]$. (b) *Subgraphs* $G[U_0]$, $G(U_1)$, *and* $G(U_2)$ *of* G.

Property R1. $U_0 \subseteq V_0$.

Property R2. For each index $i = 1, \ldots, p$, there is an index j with $1 \le j \le q$ and $V_i \subseteq U_j$.

Property R3. For any indices i, i', i'' with $1 \le i < i' < i'' \le p$, if $V_i \cup V_{i''} \subseteq U_j$, then $V_{i'} \subseteq U_j$.

For instance, in Figure 4(a), $[V_0, V_1, V_2, V_3]$ is a refinement of $[U_0, U_1, U_2]$. Below is the main lemma of the section.

LEMMA 3.1. *Let k be a positive integer. Let G be an n-node connected graph embedded on a genus-$o(n/\ell_k^2)$ surface. Given a $(k-1)$-separation $\mathbb{S}_{k-1}$ of G, it takes $O(n)$ time to compute a k-separation $\mathbb{S}_k$ of G that is a refinement of $\mathbb{S}_{k-1}$.*

The proof of Lemma 3.1 needs the following lemma, which can be proved by Lemmas 2.4 and 2.6.

LEMMA 3.2. *Let k be a positive integer. Given an n-node graph G embedded on a genus-$o(n/\ell_k^2)$ surface, it takes $O(n)$ time to compute an $o(\frac{n}{\ell_k})$-node subset V of $Node(G)$ such that each node of $Node(G) \setminus V$ has degree at most ℓ_k^2 in G and each connected component of $G \setminus V$ has at most ℓ_k^4 nodes.*

Proof. We first apply Lemma 2.6 to compute in $O(n)$ time an $o(\frac{n}{\ell_k})$-node subset V' of $Node(G)$ such that $G \setminus V'$ is planar. We then apply Lemma 2.4 to compute in $O(n)$ time a separator decomposition $\mathbb{T}$ of $G \setminus V'$. For each vertex S of $\mathbb{T}$, let *Offspring*(S) denote the union of all the vertices in the subtree of $\mathbb{T}$ rooted at S, and let *offspring*$(S) = |$*Offspring*$(S)|$. Let $r = \ell_k^2$. Let V'' consist of the nodes of G with degree more than r in G. Let V''' be the union of all the vertices S of $\mathbb{T}$ with *offspring*$(S) > r^2$. Let $V = V' \cup V'' \cup V'''$. By $V' \cup V''' \subseteq V$ and the definition of $\mathbb{T}$, each connected component of $G \setminus V$ has at most r^2 nodes. By $V'' \subseteq V$, each node of $Node(G) \setminus V$ has degree at most r in G. Since G has $O(n)$ edges, $|V''| = O(\frac{n}{r}) = o(\frac{n}{\ell_k})$. It remains to show that $|V'''| = o(\frac{n}{\ell_k})$. For each index $i \ge 1$, let $\mathbb{I}_i$ consist of the vertices S of $\mathbb{T}$ with $r^2 \cdot (\frac{3}{2})^{i-1} <$ *offspring*$(S) \le r^2 \cdot (\frac{3}{2})^i$. By $r^2 \ge 1$ and $i \ge 1$, each $S \in \mathbb{I}_i$ is an internal vertex of $\mathbb{T}$. By definition of $\mathbb{T}$, we know that *Offspring*(S) and *Offspring*(S') are disjoint for any two distinct elements S and S' of $\mathbb{I}_i$, implying that $\sum_{S \in \mathbb{I}_i}$ *offspring*$(S) \le n$ holds. Since *offspring*$(S) > r^2 \cdot (1.5)^{i-1}$ holds for each $S \in \mathbb{I}_i$, we have $|\mathbb{I}_i| < \frac{n}{r^2 \cdot (1.5)^{i-1}}$. Since each $S \in \mathbb{I}_i$ is an internal vertex of $\mathbb{T}$, S is a separator of $G[$*Offspring*$(S)]$. Therefore, $|S| = O(r \cdot (1.5)^{i/2})$ holds for each vertex S in $\mathbb{I}_i$. We have $|V'''| = \sum_{i \ge 1} \sum_{S \in \mathbb{I}_i} |S| = \sum_{i \ge 1} O(\frac{n}{r \cdot (1.5)^{i/2}}) = O(\frac{n}{r}) = o(\frac{n}{\ell_k})$. The lemma is proved. ∎

Algorithm 1

Let $p = 0$, and let all elements of $\mathbb{C}$ be initially unmarked.
For each $j = 1, \ldots, q$, perform the following repeat-loop.
Repeat the following steps until all elements of $\mathbb{C}_j$ are marked:
Let v_0 be an arbitrary node of V_0 adjacent to some unmarked element of $\mathbb{C}_j$.
Let $\mathbb{U}$ consist of the unmarked elements of $\mathbb{C}_j$ that are adjacent to v_0 in G.
Let $C_{i_1}, \ldots, C_{i_3}$ be the elements of $\mathbb{U}$ in clockwise order around v_0 in G.
Mark all $i_3 - i_1 + 1$ elements of $\mathbb{U}$.
Repeat the following four steps until $i_1 > i_3$:
Let i_2 be the largest index with $i_1 \le i_2 \le i_3$ and $|C_{i_1}| + \cdots + |C_{i_2}| \le \ell_k^4$.
Let $p = p + 1$.
Let $hook_p = v_0$ and $V_p = C_{i_1} \cup \cdots \cup C_{i_2}$.
Let $i_1 = i_2 + 1$.
Output $V_1, \ldots, V_p$ and $hook_1, \ldots, hook_p$.

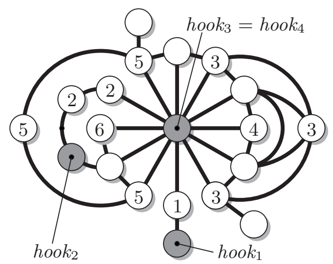
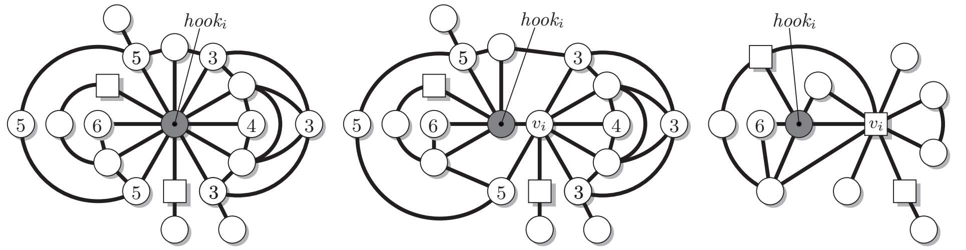


FIG. 5. *An illustration for Algorithm* 1.

Proof of Lemma 3.1. Suppose that $[U_0, \ldots, U_q]$ is the given $(k-1)$-separation $\mathbb{S}_{k-1}$. Let V_0' be the $O(n)$-time computable subset of $Node(G)$ ensured by Lemma 3.2. We have $|V_0'| = o(\frac{n}{\ell_k})$. Let $V_0 = U_0 \cup V_0'$. Let $\mathbb{C}$ consist of the connected components of $G \setminus V_0$. By $V_0' \subseteq V_0$, each element of $\mathbb{C}$ has at most ℓ_k^4 nodes. By $U_0 \subseteq V_0$ and Properties S1 and S2 of $\mathbb{S}_{k-1}$, each element of $\mathbb{C}$ is contained by some U_j with $1 \le j \le q$. For each $j = 1, \ldots, q$, let $\mathbb{C}_j$ consist of the elements C of $\mathbb{C}$ with $C \subseteq U_j$. We run Algorithm 1 to obtain (a) a disjoint partition $V_1, \ldots, V_p$ of $G \setminus V_0$ and (b) p nodes $hook_1, \ldots, hook_p$ of V_0, which may not be distinct. Let $\mathbb{S}_k = [V_0, \ldots, V_p]$. Since G is connected, each element of $\mathbb{C}$ is adjacent to V_0. The first statement of the outer repeat-loop is well defined. Since each element of $\mathbb{C}$ has at most ℓ_k^4 nodes, the first statement of the inner repeat-loop is well defined. See Figure 5 for an illustration: Suppose that all nodes are in U_1. All nodes are initially unmarked. Let V_0 consist of the nine unlabeled nodes, including the three gray nodes. For each $i = 1, \ldots, 6$, let C_i consist of the nodes with label i. That is, $C_1, \ldots, C_6$ are the six connected components of $G \setminus V_0$. Suppose that $\ell_k^4 = 7$ and the first two iterations of the outer repeat-loop obtain $V_1 = C_1$ and $V_2 = C_2$. In the third iteration of the outer repeat-loop, $C_3, \ldots, C_6$ are the unmarked elements of $\mathbb{C}$ that are adjacent to $hook_3$ in clockwise order around $hook_3$. By $|C_3| + |C_4| + |C_5| = 7$, the two iterations of the inner repeat-loop obtain $V_3 = C_3 \cup C_4 \cup C_5$ and $V_4 = C_6$.

By definition of Algorithm 1, one can verify that Properties R1, R2, and R3 hold for $\mathbb{S}_{k-1}$ and $\mathbb{S}_k$ (that is, $\mathbb{S}_k$ is a refinement of $\mathbb{S}_{k-1}$) and Properties S1 and S2 hold for $\mathbb{S}_k$. By Property S3 of $\mathbb{S}_{k-1}$, we have $|U_0| = o(\frac{n}{\ell_{k-1}}) = o(\frac{n}{\ell_k})$. By $|V_0'| = o(\frac{n}{\ell_k})$,



(a) (b) (c)

FIG. 6. *The operation that contracts all nodes of V_i into a node v_i, which takes over some neighbors of $hook_i$.*

we have $|V_0| \leq |U_0| + |V'_0| = o(\frac{n}{\ell_k})$. Let I_{small} consist of the indices i with $1 \leq i \leq p$ and $|V_i| \leq \frac{1}{2} \cdot \ell_k^4$. Let I_{large} consist of the indices i with $1 \leq i \leq p$ and $|V_i| > \frac{1}{2} \cdot \ell_k^4$. We show that $p = |I_{\text{small}}| + |I_{\text{large}}| = o(\frac{n}{\ell_k})$ as follows. By Property S1 of $\mathbb{S}_k$, we have $|I_{\text{large}}| = o(\frac{n}{\ell_k})$. To show that $|I_{\text{small}}| = o(\frac{n}{\ell_k})$, we categorize the indices i in I_{small} with $1 \leq i < p$ into the the following types, where j is the index with $V_i \subseteq U_j$:

Type 1: $i \in I_{\text{small}}$ and $i+1 \in I_{\text{large}}$. The number of such indices i is no more than $|I_{\text{large}}| = o(\frac{n}{\ell_k})$.

Type 2: $i \in I_{\text{small}}$ and $i+1 \in I_{\text{small}}$.

Type 2a: $V_{i+1} \subseteq U_{j+1}$. The number of such indices i is no more than $q = o(\frac{n}{\ell_{k-1}}) = o(\frac{n}{\ell_k})$.

Type 2b: $V_{i+1} \subseteq U_j$ and $hook_i \in V_0 \setminus U_0$. By Properties S1 and S2 of $\mathbb{S}_{k-1}$, we know that $hook_i \in U_j$. By definition of Algorithm 1, $hook_{i'} \neq hook_i$ holds for all indices i' with $i < i' \leq p$. The number of such indices i is no more than $|V_0 \setminus U_0| \leq |V_0| = o(\frac{n}{\ell_k})$.

Type 2c: $V_{i+1} \subseteq U_j$ and $hook_i \in U_0$. We have $hook_i \in Nbr_G(U_j)$. By definition of Algorithm 1, $hook_{i'} \neq hook_i$ holds for all indices $i' > i$ with $V_{i'} \subseteq U_j$. By Property S5 of $\mathbb{S}_{k-1}$, the number of such indices i is no more than $\sum_{j=1}^{q} nbr_G(U_j) = o(\frac{n}{\ell_{k-1}}) = o(\frac{n}{\ell_k})$.

We have $p = o(\frac{n}{\ell_k})$. Property S3 holds for $\mathbb{S}_k$. By definition of Algorithm 1, $|V_i| \leq \ell_k^4$ holds for each $i = 1, \ldots, p$. By $V'_0 \subseteq V_0$, each node of $Node(G) \setminus V_0$ has degree at most ℓ_k^2. Property S4 holds for $\mathbb{S}_k$.

To see Property S5 of $\mathbb{S}_k$, we obtain a contracted graph from G by performing the following two steps for each $i = 1, \ldots, p$.[6] *Step* 1: Let $C_{i_1}, \ldots, C_{i_2}$ be the elements of $\mathbb{C}$ with $V_i = C_{i_1} \cup C_{i_1+1} \cup \cdots \cup C_{i_2}$ in clockwise order around $hook_i$ in G. Split $hook_i$ into two adjacent nodes $hook_i$ and v_i, and let v_i take over the neighbors of $hook_i$ in clockwise order around $hook_i$ from the first neighbor of $hook_i$ in C_{i_1} to the first neighbor of $hook_i$ in C_{i_2}. *Step* 2: Contract all nodes of V_i into node v_i, and delete multiple edges and self-loops. See Figure 6 for an illustration: For each $i = 3, \ldots, 6$, let C_i consist of the nodes with labels i in Figure 6(a). Suppose that $i_1 = 3$, $i_2 = 5$, and $V_i = C_3 \cup C_4 \cup C_5$. The unlabeled circle nodes belong to V_0. The square nodes are two previously contracted nodes $v_{i'}$ and $v_{i''}$ from $V_{i'}$ and $V_{i''}$ for some indices i' and i'' with $1 \leq i' \neq i'' < i$. Figure 6(b) shows the result of Step 1. Figure 6(c) shows the result of Step 2. Observe that each node that is adjacent to V_i becomes a

[6]The contraction procedure is only for proving Property S5 of $\mathbb{S}_k$; it is not needed for computing $\mathbb{S}_k$.

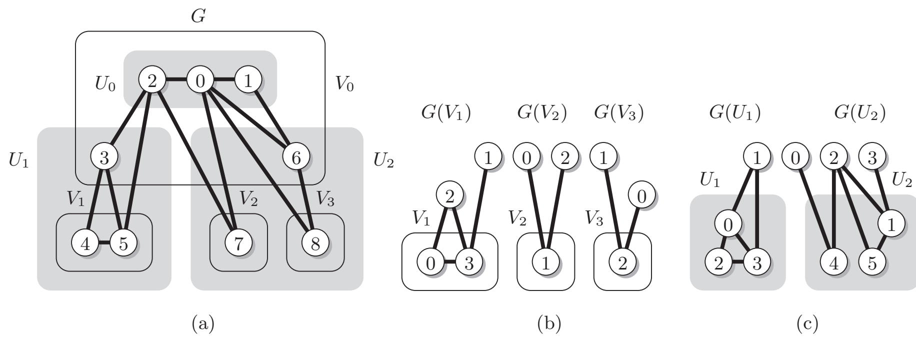


FIG. 7. (a) *Graph G with a labeling.* (b) *Subgraphs $G(V_1)$, $G(V_2)$, and $G(V_3)$ of G with labelings.* (c) *Subgraphs $G(U_1)$ and $G(U_2)$ of G with labelings.*

neighbor of v_i after applying Steps 1 and 2. Also, each neighbor of $hook_i$ that is not in V_i either remains a neighbor of $hook_i$ or becomes a neighbor of v_i after applying Steps 1 and 2. Therefore, for each $i = 1, \ldots, p$ and each node $v_0 \in Nbr_G(V_i)$, there is either an edge (v_0, v_i) or an edge $(v_i, v_{i'})$ for some index i' with $i' > i$ and $hook_{i'} = v_0$. Thus, $\sum_{i=1}^{p} nbr_G(V_i)$ is no more than the number of edges in the resulting contracted simple graph, which has $|V_0| + p = o(\frac{n}{\ell_k})$ nodes. Observe that Step 1 does not increase the genus of the embedding. Since the subgraph induced by $V_i \cup \{v_i\}$ is connected, Step 2 does not increase the genus of the embedding either. The number of edges in the resulting contracted simple genus-$o(n/\ell_k^2)$ graph is $o(\frac{n}{\ell_k})$. Property S5 holds for $\mathbb{S}_k$. The lemma is proved. $\square$

4. Our compression scheme. This section proves Theorem 1.1.

4.1. Recovery string. A *labeling* of graph G is a one-to-one mapping from $Node(G)$ to $\{0, 1, \ldots, node(G) - 1\}$. For instance, Figure 7(a) shows a labeling for graph G. Let G be a graph embedded on a surface. We say that a graph Δ embedded on the same surface is a *triangulation* of G if G is a subgraph of Δ with $Node(\Delta) = Node(G)$ such that each face of Δ has three nodes. The following lemma shows an $o(n)$-bit string with which the larger embedded labeled subgraphs of G can be recovered from smaller embedded labeled subgraphs of G in $O(n)$ time.

LEMMA 4.1. *Let k be a positive integer. Let G be an n-node graph embedded on a genus-$o(\frac{n}{\ell_k})$ surface. Let Δ be a triangulation of G. Let $\mathbb{S}_k = [V_0, \ldots, V_p]$ be a given k-separation of Δ and $\mathbb{S}_{k-1} = [U_0, \ldots, U_q]$ be a given $(k-1)$-separation of Δ such that $\mathbb{S}_k$ is a refinement of $\mathbb{S}_{k-1}$. For any given labeling $L_{k,i}$ of $G(V_i)$ for each $i = 1, \ldots, p$, the following statements hold*:

(1) *It takes overall $O(n)$ time to compute a labeling $L_{k-1,j}$ of subgraph $G(U_j)$ for each $j = 1, \ldots, q$.*
(2) *Given the above labelings $L_{k-1,j}$ of subgraphs $G(U_j)$ with $1 \le j \le q$, it takes $O(n)$ time to compute an $o(n)$-bit string Rec_k such that $G(U_j)$ and $L_{k-1,j}$ for all $j = 1, \ldots, q$ can be recovered in overall $O(n)$ time from Rec_k and $G(V_i)$ and $L_{k,i}$ for all $i = 1, \ldots, p$.*

Proof. Since Δ is a subgraph G with $Node(\Delta) = Node(G)$, one can easily verify that $\mathbb{S}_{k-1}$ (respectively, $\mathbb{S}_k$) is also a $(k-1)$-separation (respectively, k-separation) of G. For each $j = 1, \ldots, q$, let I_j consist of the indices i with $V_i \subseteq U_j$. Let W_j consist of the nodes of $G(U_j)$ that are not in any V_i with $i \in I_j$. By Properties S1 and S2 of $\mathbb{S}_k$, $W_j \subseteq V_0$. For instance, if G is as shown in Figure 7(a), where v_t with $0 \le t \le 8$

denotes the node with label t, we have $I_1 = \{1\}$, $I_2 = \{2,3\}$, $W_1 = \{v_2, v_3\}$, and $W_2 = \{v_0, v_1, v_2, v_6\}$. Let the labeling $L_{k-1,j}$ for $G(U_j)$ be defined as follows:

- For the nodes of $G(U_j)$ in W_j, let $L_{k-1,j}$ be an arbitrary one-to-one mapping from W_j to $\{0, 1, \ldots, |W_j| - 1\}$. In Figure 7(c), we have $L_{k-1,1}(v_2) = 1$, $L_{k-1,1}(v_3) = 0$, $L_{k-1,2}(v_0) = 2$, $L_{k-1,2}(v_1) = 3$, $L_{k-1,2}(v_2) = 0$, and $L_{k-1,2}(v_6) = 1$.
- For the nodes of $G(U_j)$ not in W_j, let $L_{k-1,j}$ be the one-to-one mapping from $\bigcup_{i \in I_j} V_i$ to $\{|W_j|, |W_j| + 1, \ldots, \mathit{node}(G(U_j)) - 1\}$ obtained by sorting $(i, L_{k,i}(v))$ for all indices $i \in I_j$ and all nodes $v \in V_i$ such that $L_{k-1,j}(v) < L_{k-1,j}(v')$ holds for a node v of V_i and a node v' of $V_{i'}$ if and only if (a) $i < i'$ or (b) $i = i'$ and $L_{k,i}(v) < L_{k,i'}(v')$. For instance, if $L_{k,1}$, $L_{k,2}$, and $L_{k,3}$ are as shown in Figure 7(b), then $L_{k-1,1}$ and $L_{k-1,2}$ can be as shown in Figure 7(c) and $L_{k-2,1}$ can be as shown in Figure 7(a).

It takes $O(\mathit{node}(G(U_j))) = O(|U_j| + \mathit{nbr}_G(U_j))$ time to compute $L_{k-1,j}$ from all $L_{k,i}$ with $i \in I_j$. By Property S5 of $\mathbb{S}_{k-1}$, it takes overall $O(n)$ time to compute all $L_{k-1,j}$ with $1 \le j \le q$ from all $L_{k,i}$ with $1 \le i \le p$. Statement (1) is proved.

By Property S4 of $\mathbb{S}_{k-1}$, the label of each node of $G(U_j)$ assigned by $L_{k-1,j}$ can be represented by $O(\log \operatorname{poly}(\ell_{k-1})) = O(\ell_k)$ bits. By Property S4 of $\mathbb{S}_k$, the label of each node of $G(V_i)$ assigned by $L_{k,i}$ can be represented by $O(\log \operatorname{poly}(\ell_k)) = O(\ell_{k+1})$ bits. For each index $j = 1, \ldots, q$,

- string $\mathit{Rec}'_{k,j}$ stores the adjacency list of the embedded subgraph of $G(V_j)$ induced by W_j via the labeling $L_{k-1,j}$ of W_j,
- string $\mathit{Rec}''_{k,j}$ stores the information required to recover $L_{k-1,j}$ from all $L_{k,i}$ with $i \in I_j$, and
- string $\mathit{Rec}'''_{k,j}$ stores the information required to recover the embedding of $G(U_j)$ from the embeddings of all $G(V_i)$ with $i \in I_j$ and the embedding of the subgraph of $G(U_j)$ induced by W_j.

By definition of W_j, we have $|W_j| = |V_0 \cap U_j| + \mathit{nbr}_G(U_j)$. It follows from Property S3 of $\mathbb{S}_k$ and Property S5 of $\mathbb{S}_{k-1}$ that

$$\sum_{j=1}^{q} |W_j| \le |V_0| + \sum_{j=1}^{q} \mathit{nbr}_G(U_j) = o\left(\frac{n}{\ell_k}\right) + o\left(\frac{n}{\ell_{k-1}}\right) = o\left(\frac{n}{\ell_k}\right).$$

Let $W = \bigcup_{j=1}^{q} W_j$. Since $G[V_0], G(V_1), \ldots, G(V_p)$ form a disjoint partition of the edges of G, the overall number of edges in the subgraphs of $G(V_j)$ induced by W_j for all $j = 1, \ldots, q$ is no more than the number of edges in $G[W]$, which is $O(|W| + o(\frac{n}{\ell_k})) \le O(\sum_{j=1}^{q} |W_j|) + o(\frac{n}{\ell_k}) = o(\frac{n}{\ell_k})$. Therefore,

$$\sum_{j=1}^{q} \|\mathit{Rec}'_{k,j}\| = o\left(\frac{n}{\ell_k}\right) \cdot O(\ell_k) = o(n). \tag{1}$$

It suffices for $\mathit{Rec}''_{k,j}$ to store the list of $(i, L_{k,i}(v), L_{k-1,j}(v))$ for all $i \in I_j$ and all $v \in \mathit{Nbr}_G(V_i)$. By Property R3 of $\mathbb{S}_{k-1}$ and $\mathbb{S}_k$ and Property S4 of $\mathbb{S}_{k-1}$, index i can be represented by an $O(\ell_k)$-bit offset t such that i is the tth smallest index in I_j. Thus, $\|\mathit{Rec}''_{k,j}\| = \sum_{i \in I_j} \mathit{nbr}_G(V_i) \cdot O(\ell_k)$. By Property S5 of $\mathbb{S}_k$, we have $\sum_{j=1}^{q} \sum_{i \in I_j} \mathit{nbr}_G(V_i) = \sum_{i=1}^{p} \mathit{nbr}_G(V_i) = o(\frac{n}{\ell_k})$. Therefore,

$$\sum_{j=1}^{q} \|\mathit{Rec}''_{k,j}\| = o\left(\frac{n}{\ell_k}\right) \cdot O(\ell_k) = o(n). \tag{2}$$

It suffices for $Rec'''_{k,j}$ to store the list of $(L_{k-1,j}(v), L_{k-1,j}(v'), L_{k-1,j}(v''))$ for all pairs of edges (v, v') and (v, v'') of $G(U_j)$ such that (a) v'' is the neighbor of v that immediately succeeds v' in clockwise order around v in $G(U_j)$ and (b) nodes v' and v'' are not in the same partition of $Node(G(U_j))$ formed by the $|I_j| + 1$ disjoint sets W_j and V_i with $i \in I_j$. By Property S2 of $\mathbb{S}_k$, node v belongs to $W_j \subseteq V_0$. Since Δ is a triangulation of G, the neighbors of v in Δ form a cycle that surrounds v in Δ. Let P be the path of the cycle from v' to v'' in clockwise order around v. At least one node, say, u, of P belongs to V_0, since otherwise Property S2 of $\mathbb{S}_k$ would imply that all nodes of P belong to the same V_i for some $i \in I_j$, contradicting the choices of v' and v''. Edge (v, u) belongs to $\Delta[V_0]$. Observe that each edge of $\Delta[V_0]$ can be identified by at most four such edge pairs (v, v') and (v, v''). Since the edges of $G(U_j)$ and $G(U_{j'})$ with $1 \le j \ne j' \le q$ are disjoint, the number of edge pairs stored in $Rec'''_{k,j}$ is at most four times the number of edges in $\Delta[V_0]$. By Property S3 of $\mathbb{S}_k$ and the fact that Δ has genus $o(\frac{n}{\ell_k})$, the number of edge pairs stored in Rec''' is $o(\frac{n}{\ell_k})$. Therefore,

$$\sum_{j=1}^{q} \|Rec'''_{k,j}\| = o\left(\frac{n}{\ell_k}\right) \cdot O(\ell_k) = o(n). \tag{3}$$

Let

$$\begin{aligned} Rec'_k &= Rec'_{k,1} \circ \cdots \circ Rec'_{k,q}, \\ Rec''_k &= Rec''_{k,1} \circ \cdots \circ Rec''_{k,q}, \\ Rec'''_k &= Rec'''_{k,1} \circ \cdots \circ Rec'''_{k,q}, \\ Rec_k &= Rec'_k \circ Rec''_k \circ Rec'''_k. \end{aligned}$$

By (1), (2), and (3) and Lemma 2.2, we have $\|Rec_k\| = o(n)$. It takes $O(n)$ time to compute Rec_k from all labelings $L_{k,j}$ and all embedded graphs $G(U_j)$ with $1 \le j \le q$ and all labelings $L_{k-1,i}$ and all embedded graphs $G(V_i)$ with $1 \le i \le p$. It also takes $O(n)$ time to recover all labelings $L_{k,j}$ and all embedded graphs $G(U_j)$ with $1 \le j \le q$ from Rec_k and all labelings $L_{k-1,i}$ and all embedded graphs $G(V_i)$ with $1 \le i \le p$. Statement (2) holds. The lemma is proved. ∎

4.2. Proving Theorem 1.1. We are ready to prove the main theorem of the paper.

Proof of Theorem 1.1. Let $G \in \mathbb{G}$ be the n-node input graph embedded on a genus-$o(\frac{n}{\log^2 n})$ surface. The encoding algorithm $Encode_A$ performs the following four steps on G:

- E1: Triangulate the embedded graph G into a triangulation Δ of G. Let $\mathbb{S}_0$ be the 0-separation $[\varnothing, Node(\Delta)]$ of Δ. For each $k = 1, 2$, apply Lemma 3.1 to obtain a k-separation $\mathbb{S}_k$ of Δ that is a refinement of $\mathbb{S}_{k-1}$.
- E2: Let $[V_0, \ldots, V_p] = \mathbb{S}_2$. Apply Lemma 2.3 with $\ell = \max_{1\le i\le p} node(G(V_i))$ to compute (a) $Label(H)$ and $Optcode(H)$ for all distinct graphs H in class $\mathbb{G}$ with $node(H) \le \ell$ and (b) $Table(\mathbb{G}, \ell)$. For each $i = 1, \ldots, p$, apply Lemma 2.3(1) to compute from $Table(\mathbb{G}, \ell)$ the binary string $Code(V_i) = Optcode(G(V_i))$ and the labeling $L_{2,i} = Label(G(V_i))$.
- E3: For each $k = 2, 1$, perform the following two substeps:
  - E3.1: Let $[U_0, \ldots, U_q] = \mathbb{S}_{k-1}$ and $[V_0, \ldots, V_p] = \mathbb{S}_k$. For each $j = 1, \ldots, q$, let binary string $Code(U_j) = Code(V_{i_1}) \circ \cdots \circ Code(V_{i_2})$, where $\{i_1, i_1 + 1, \ldots, i_2\}$ are the indices i with $V_i \subseteq U_j$.

E3.2: Apply Lemma 4.1(1) to obtain the labelings $L_{k-1,j}$ of subgraphs $G(U_j)$ for all $j = 1, \ldots, q$. Apply Lemma 4.1(2) to obtain the $o(n)$-bit binary string Rec_k.

E4: By $\mathbb{S}_0 = [\varnothing, Node(G)]$, now we have $Code(Node(G))$ (and a labeling $L_{0,1}$ for $G = G(Node(G)))$. The output binary string $Code_A(G) = Code(Node(G)) \circ Table(\mathbb{G}, \ell) \circ Rec_1 \circ Rec_2$.

By Lemma 3.1, Step E1 takes $O(n)$ time. By Property S5 of $\mathbb{S}_2$, $\sum_{i=1}^{p} node(G(V_i)) = n + o(n)$. By Lemma 2.3, Step E2 takes $O(n)$ time. By Lemmas 2.2 and 4.1, Step E3 takes $O(n)$ time. By Lemma 2.2, Step E4 takes $O(n)$ time. Therefore, the encoding algorithm $Encode_A(G)$ runs in $O(n)$ time. Condition C1 holds.

The decoding algorithm $Decode_A$ performs the following five steps on $Code_A(G)$:

D1: Obtain $Code(Node(G))$, $Table(\mathbb{G}, \ell)$, Rec_1, and Rec_2 from $Code_A(G)$.

D2: Let $\mathbb{S}_0 = [\varnothing, Node(G)]$. For each $k = 1, 2$, perform the following substep:

D2.1: Let $[U_0, \ldots, U_q] = \mathbb{S}_{k-1}$ and $[V_0, \ldots, V_p] = \mathbb{S}_k$. For each $j = 1, \ldots, q$, obtain all $Code(V_i)$ with $V_i \subseteq U_j$ from $Code(U_j)$.

D3: Let $[V_0, \ldots, V_p] = \mathbb{S}_2$. For each $i = 1, \ldots, p$, apply Lemma 2.3(2) to obtain $G(V_i)$ and $L_{2,i} = Label(G(V_i))$ from $Code(V_i) = Optcode(G(V_i))$ and $Table(\mathbb{G}, \ell)$.

D4: For each $k = 2, 1$, perform the following substep:

D4.1: Let $[U_0, \ldots, U_q] = \mathbb{S}_{k-1}$ and $[V_0, \ldots, V_p] = \mathbb{S}_k$. Apply Lemma 4.1(2) to recover $G(U_j)$ and $L_{k-1,j}$ with $1 \le j \le q$ from $G(V_i)$ and $L_{k,i}$ with $1 \le i \le p$ and Rec_k.

D5: Output $G = G(Node(G))$.

By Lemma 2.2, Step D1 takes $O(n)$ time. By Lemma 2.2, Step D2 takes $O(n)$ time. By Property S5 of $\mathbb{S}_2$, we have $\sum_{i=1}^{p} node(G(V_i)) = n + o(n)$. By Lemma 2.3(2), Step D3 takes $O(n)$ time. By Lemma 4.1(2), Step D4 takes $O(n)$ time. Therefore, the decoding algorithm $Decode_A(G)$ runs in $O(n)$ time. Condition C2 holds. By $\mathbb{S}_0 = [\varnothing, Node(G)]$, graph $G = G(Node(G))$ is correctly recovered from $Code_A(G)$ at the end of Step D4. Therefore, A is a compression scheme for $\mathbb{G}$.

To show Condition C3, we first prove the following claim for each $k = 1, 2$.

Claim 1. *Suppose that* $[U_0, \ldots, U_q] = \mathbb{S}_{k-1}$ *and* $[V_0, \ldots, V_p] = \mathbb{S}_k$. *If*

$$\sum_{i=1}^{p} \|Code(V_i)\| \le \beta n + o(n)$$

and $\|Code(V_i)\| = poly(\ell_k)$ *holds for each* $i = 1, \ldots, p$, *then* $\sum_{j=1}^{q} \|Code(U_j)\| \le \beta n + o(n)$ *and* $\|Code(U_j)\| = poly(\ell_{k-1})$ *holds for each* $j = 1, \ldots, q$.

Proof of Claim 1. For each $j = 1, 2, \ldots, q$, let I_j consist of the indices i with $V_i \subseteq U_j$. By Property S4 of $\mathbb{S}_{k-1}$, we have $|I_j| \le |U_j| = \mathrm{poly}(\ell_{k-1})$. Therefore, $\sum_{i \in I_j} \|Code(V_i)\| = \mathrm{poly}(\ell_{k-1})$, implying $O(\log \sum_{i \in I_j} \|Code(V_i)\|) = O(\ell_k)$. By Property S3 of $\mathbb{S}_k$, $\sum_{j=1}^{q} |I_j| = p = O(\frac{n}{\ell_k})$. By Lemma 2.2, we have

$$\sum_{j=1}^{q} \|Code(U_j)\| = \sum_{j=1}^{q} O(|I_j| \cdot \ell_k) + \sum_{i=1}^{p} \|Code(V_i)\| \le \beta n + o(n).$$

We also have

$$\|Code(U_j)\| = O(|I_j| \cdot \ell_k) + \sum_{i \in I_j} \|Code(V_i)\| = \mathrm{poly}(\ell_{k-1}).$$

The claim is proved. $\square$

Let $[V_0,\ldots,V_p] = \mathbb{S}_2$. For each $i = 1,\ldots,p$, let $n_i = \mathit{node}(G(V_i)) = |V_i| + \mathit{nbr}_G(V_i)$. By Property S5 of $\mathbb{S}_2$, $\sum_{i=1}^{p} n_i = n + o(n)$. By Step E2 of $\mathit{Encode}_A(G)$ and Lemma 2.3, we have $\|\mathit{Code}(V_i)\| = \|\mathit{Optcode}(G(V_i))\| = \lceil \log \mathit{num}(\mathbb{G}, n_i)\rceil \le \beta n_i + o(n_i)$. By Property S4 of $\mathbb{S}_2$ and the assumption that $\log \mathit{num}(\mathbb{G}, n) = O(n)$,

$$\|\mathit{Code}(V_i)\| = \mathrm{poly}(\ell_2) \tag{4}$$

holds for each $i = 1, 2, \ldots, p$. We have

$$\sum_{i=1}^{p} \|\mathit{Code}(V_i)\| \le \sum_{i=1}^{p} (\beta n_i + o(n_i)) = \beta\cdot(n + o(n)) + o(\beta n + o(n)) = \beta n + o(n). \tag{5}$$

By combining (4) and (5), Claim 1 for $k = 2, 1$, and $\mathbb{S}_0 = [\varnothing, \mathit{Node}(G)]$, we have that $\|\mathit{Code}(\mathit{Node}(G))\| \le \beta n + o(n)$. By Lemma 2.2 and $\|\mathit{Table}(\mathbb{G}, \ell)\| + \|\mathit{Rec}_1\| + \|\mathit{Rec}_2\| = o(n)$, we have that $\|\mathit{Code}_A(G)\| \le \beta n + o(n)$. Condition C3 holds. The theorem is proved. $\square$

5. Extension. This section proves Theorem 1.2. The only place in our proof of Theorem 1.1 requiring $\mathbb{G}$ to be hereditary is Step E2: We need $G(V_i) \in \mathbb{G}$ so that $\mathit{Optcode}(G(V_i))$ and $\mathit{Label}(G(V_i))$ can be obtained from $\mathit{Table}(\mathbb{G}, \ell)$. For a nonhereditary class $\mathbb{G}$, we can substitute $G(V_i)$ by a graph $H_i \in \mathbb{G}$ that is close to $G(V_i)$ for each $i = 1, 2, \ldots, p$ as long as the overall number of bits required to encode the overall difference between $G(V_i)$ and H_i is $o(n)$. The following corollary is an example of such an extension.

Corollary 5.1. *Let $\mathbb{G}$ be a class of graphs satisfying $\log \mathit{num}(\mathbb{G}, n) = O(n)$ and such that any input n-node graph $G \in \mathbb{G}$ to be encoded comes with a genus-$o(\frac{n}{\log^2 n})$ embedding. If, for any 2-separation $[V_0,\ldots,V_p]$ of any graph $G \in \mathbb{G}$, there exist graphs $H_1,\ldots,H_p$ in $\mathbb{G}$ such that each $G(V_i)$ with $1 \le i \le p$ can be obtained from H_i by first deleting $O(\mathit{nbr}_G(V_i))$ nodes (together with their incident edges) and then updating (adding or deleting) $O(\mathit{nbr}_G(V_i))$ edges, then $\mathbb{G}$ admits an optimal compression scheme.*

Proof. We revise algorithm Encode_A by updating Steps E2 and E4 as follows:

- E2′: Let $[V_0,\ldots,V_p] = \mathbb{S}_2$. Compute $H_1,\ldots,H_p$ from $G(V_1),\ldots,G(V_p)$. With $\ell = \max_{1\le i\le p} \mathit{node}(H_i)$, apply Lemma 2.3 to compute (a) $\mathit{Label}(H)$ and $\mathit{Optcode}(H)$ for each distinct graph $H \in \mathbb{G}$ with $\mathit{node}(H) \le \ell$ and (b) $\mathit{Table}(\mathbb{G}, \ell)$. Apply Lemma 2.3(1) to compute $\mathit{Code}(V_i) = \mathit{Optcode}(H_i)$ and $L'_{2,i} = \mathit{Label}(H_i)$ from $\mathit{Table}(\mathbb{G}, \ell)$ for all indices $i = 1,\ldots,p$. Let $L_{2,i}$ be the labeling of $G(V_i)$ obtained from the labeling $L'_{2,i}$ of H_i such that if v and v' are two distinct nodes of $G(V_i)$ with $L'_{2,i}(v) < L'_{2,i}(v')$, then we have $L_{2,i}(v) < L_{2,i}(v')$. Let Fix_i be the binary string storing the difference between H_i and $G(V_i)$ via labeling $L'_{2,i}$. Let $\mathit{Fix} = \mathit{Fix}_1 \circ \cdots \circ \mathit{Fix}_p$.
- E4′: By $\mathbb{S}_0 = [\varnothing, \mathit{Node}(G)]$, now we have $\mathit{Code}(\mathit{Node}(G))$ (and a labeling $L_{0,1}$ for $G = G(\mathit{Node}(G))$). The output binary string $\mathit{Code}_A(G)$ for G is $\mathit{Code}(\mathit{Node}(G)) \circ \mathit{Table}(\mathbb{G}, \ell) \circ \mathit{Rec}_1 \circ \mathit{Rec}_2 \circ \mathit{Fix}$.

By $O(1)^{\mathrm{poly}(\ell)} = o(n)$, it takes $o(n)$ time to compute an $o(n)$-bit string Table' such that graphs $H_1, H_2, \ldots, H_p$ that satisfy the above conditions can be obtained from $G(V_1), G(V_2), \ldots, G(V_p)$ and Table' in $O(n)$ time. By Property S5 of $\mathbb{S}_2$ and the conditions of $H_1,\ldots,H_p$, we have $\sum_{i=1}^{p} \mathit{node}(G(V_i)) \le \sum_{i=1}^{p} \mathit{node}(H_i) = n + o(n)$. By Lemmas 2.2 and 2.3, Step E2′ takes $O(n)$ time. By Lemma 2.2, Step E4′ takes $O(n)$

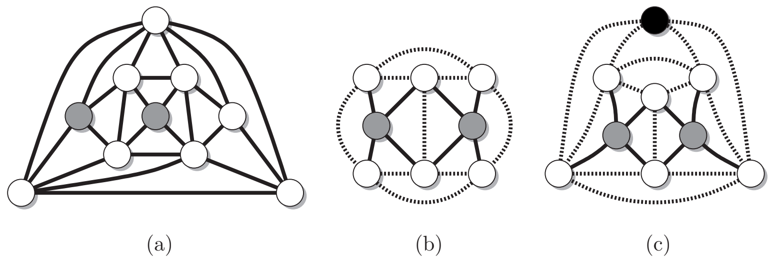


FIG. 8. (a) *A plane triangulation* Δ*, where* V_i *consists of the gray nodes.* (b) *The subgraph* $\Delta[V_i]$*, where the dotted edges are those in* $\Delta[V_i] \setminus \Delta(V_i)$*.* (c) *A plane triangulation* H_i *obtained by adding the dark node* v_F *and four edges in the external face* F *of* H_i*.*

time. Therefore, Condition C1 holds for the revised $Encode_A$. We revise algorithm $Decode_A$ by updating Steps D1 and D3 as follows.

D1′: Obtain $Code(G)$, $Table(\mathbb{G}, \ell)$, Rec_1, Rec_2, and Fix from $Code_A(G)$.

D3′: Let $[V_0, \ldots, V_p] = \mathbb{S}_2$. Apply Lemma 2.3(2) to obtain H_i and $L'_{2,i} = Label(H_i)$ from $Code(V_i) = Optcode(H_i)$ and $Table(\mathbb{G}, \ell)$ for each $i = 1, \ldots, p$. Apply Lemma 2.2 to obtain all Fix_i with $1 \le i \le p$ from Fix. Obtain $G(V_i)$ and $L_{2,i}$ from Fix_i, H_i, and $L'_{2,i}$ for all $i = 1, 2, \ldots, p$.

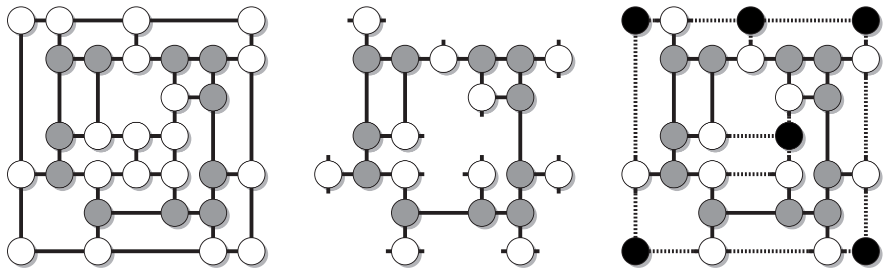
Both revised steps take $O(n)$ time. Condition C2 holds for the revised $Decode_A$. Subgraph $G(V_i)$ can be obtained from H_i by first deleting $O(nbr_G(V_i))$ nodes (and their incident edges) and then updating $O(nbr_G(V_i))$ edges. By Property S4 of $\mathbb{S}_2$, we have $\|Fix_i\| = O(nbr_G(V_i) \cdot \ell_2)$. By Property S5 of $\mathbb{S}_2$, we have $\sum_{i=1}^{p} \|Fix_i\| = o(\frac{n}{\ell_2}) \cdot O(\ell_2) = o(n)$. By Lemma 2.2, we have $\|Fix\| = o(n)$. Condition C3 holds the revised $Code_A(G)$. The corollary is proved. □

We use Corollary 5.1 to prove Theorem 1.2.

Proof of Theorem 1.2. Let Δ be an n-node triangulation of a genus-g surface. Let $[V_0, \ldots, V_p]$ be a 2-separation of Δ. Let $\mathbb{F}$ consist of the nontriangle faces of $\Delta[V_i]$. Let H_i be the plane triangulation obtained from $\Delta[V_i]$ by performing the following two steps for each face $F \in \mathbb{F}$: (1) Add a node v_F in F. (2) For each node u on the boundary of F, add an edge (u, v_F). See Figure 8 for an illustration. Since Δ is a triangulation, the boundary of F contains at least two nodes u with $Nbr_\Delta(u) \not\subseteq Node(\Delta(V_i))$. Therefore, at least two nodes of $Nbr(V_i)$ belong to the boundary of F. Let e_F be an edge between two arbitrary nodes of $Nbr(V_i)$ that belong to the boundary of F. The union of e_F over all faces $F \in \mathbb{F}$ has genus no more than $g = O(1)$. Therefore, the number of added nodes to triangulate $\Delta[V_i]$ is $O(nbr_\Delta(V_i))$. The number of edges in $\Delta[V_i] \setminus \Delta(V_i)$ is also $O(nbr_\Delta(V_i))$. Thus, $\Delta(V_i)$ can be obtained from H_i by first deleting $O(nbr_\Delta(V_i))$ nodes together with their incident edges and then deleting $O(nbr_\Delta(V_i))$ edges. By Corollary 5.1, statement (1) is proved.

Let G be an n-node floorplan. Since each node of G has at most three neighbors in G, one can easily obtain a floorplan H_i from $G(V_i)$ by adding $O(nbr_G(V_i))$ nodes and edges. See Figure 9 for an example. Statement (2) follows from Corollary 5.1. □

6. Concluding remarks. Our optimal compression schemes rely on a linear-time obtainable embedding. Can this requirement be dropped? It would be of interest to extend our compression schemes to support efficient queries and updates. We



(a) (b) (c)

FIG. 9. (a) *A floorplan G, where V_i consists of the gray nodes.* (b) *The subgraph $G(V_i)$.* (c) *A floorplan H_i obtained from $G(V_i)$ by adding $O(nbr_G(V_i))$ nodes and edges.*

leave open the problems of obtaining optimal compression schemes for $O(1)$-connected genus-$O(1)$ graphs and 3D floorplans [22, 23, 59, 60, 61, 86, 98, 99].

REFERENCES

[1] I. Abraham, D. Malkhi, and D. Ratajczak, *Compact multicast routing*, in Proceedings of the 23rd International Symposium on Distributed Computing, Lecture Notes in Comput. Sci. 5805, Springer, Berlin, 2009, pp. 364–378.

[2] M. Adler and M. Mitzenmacher, *Towards compressing web graphs*, in Proceedings of the 11th IEEE Data Compression Conference, 2001, pp. 203–212.

[3] R. Agarwal, P. B. Godfrey, and S. Har-Peled, *Approximate distance queries and compact routing in sparse graphs*, in Proceedings of the 30th IEEE International Conference on Computer Communications, 2011, pp. 1754–1762.

[4] A. R. Agnihotri, S. Ono, and P. H. Madden, *An effective approach for large scale floorplanning*, in Proceedings of the 20th ACM Great Lakes Symposium on VLSI, 2010, pp. 107–110.

[5] V. N. Anh and A. Moffat, *Local modeling for webgraph compression*, in Proceedings of the 20th IEEE Data Compression Conference, 2010, p. 519.

[6] S. R. Arikati, A. Maheshwari, and C. D. Zaroliagis, *Efficient computation of implicit representations of sparse graphs*, Discrete Appl. Math., 78 (1997), pp. 1–16.

[7] Y. Asano, Y. Miyawaki, and T. Nishizeki, *Efficient compression of web graphs*, IEICE Trans. Fund. Electron. Comm. Comput. Sci., 92-A (2009), pp. 2454–2462.

[8] P. Banerjee, S. Sur-Kolay, and A. Bishnu, *Fast unified floorplan topology generation and sizing on heterogeneous FPGAs*, IEEE Trans. CAD Integrated Circ. Syst., 28 (2009), pp. 651–661.

[9] J. Barbay, L. C. Aleardi, M. He, and J. I. Munro, *Succinct representation of labeled graphs*, in Proceedings of the 18th International Symposium on Algorithms and Computation, T. Tokuyama, ed., 2007, pp. 316–328.

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

[11] D. Benoit, E. D. Demaine, J. I. Munro, R. Raman, V. Raman, and S. S. Rao, *Representing trees of higher degree*, Algorithmica, 43 (2005), pp. 275–292.

[12] S. N. Bhatt and F. T. Leighton, *A framework for solving VLSI graph layout problems*, J. Comput. System Sci., 28 (1984), pp. 300–343.

[13] D. K. Blandford, G. E. Blelloch, and I. A. Kash, *Compact representations of separable graphs*, in Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 679–688.

[14] G. E. Blelloch and A. Farzan, *Succinct representations of separable graphs*, in Proceedings of the 21st Annual Symposium on Combinatorial Pattern Matching, A. Amir and L. Parida, eds., Lecture Notes in Comput. Sci. 6129, Springer, Berlin, 2010, pp. 138–150.

[15] A. Z. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. L. Wiener, *Graph structure in the Web*, Comput. Networks, 33 (2000), pp. 309–320.

[16] S. Chechik, *Fault-tolerant compact routing schemes for general graphs*, in Proceedings of the 38th International Colloquium on Automata, Languages and Programming, L. Aceto, M. Henzinger, and J. Sgall, eds., Lecture Notes in Comput. Sci. 6756, Springer, Berlin, 2011, pp. 101–112.

[17] T.-C. Chen and Y.-W. Chang, *Modern floorplanning based on B^*-tree and fast simulated annealing*, IEEE Trans. CAD Integrated Circ. Syst., 25 (2006), pp. 637–650.

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

[19] P.-Y. Chuang, *Improved Compact Encoding of Rectangular Drawings*, Master's thesis, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, ROC, 2008.

[20] 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, Berlin, 1998, pp. 118–129.

[21] F. Claude and G. Navarro, *Fast and compact web graph representations*, ACM Trans. Web, 4 (2010), pp. 16.1–16.31.

[22] J. Cong and Y. Ma, *Thermal-aware 3D floorplan*, in Three Dimensional Integrated Circuit Design, Y. Xie, J. Cong, and S. Sapatnekar, eds., Springer, Berlin, 2010, pp. 63–102.

[23] D. Cuesta, J. Ayala, J. Hidalgo, M. Poncino, A. Acquaviva, and E. Macii, *Thermal-aware floorplanning exploration for 3D multi-core architectures*, in Proceedings of the 20th ACM Great Lakes Symposium on VLSI 2009, R. I. Bahar, F. Lombardi, D. Atienza, and E. Brunvand, eds., 2010, pp. 99–102.

[24] P. Dasgupta and S. Sur-Kolay, *Slicible rectangular graphs and their optimal floorplans*, ACM Trans. Design Automation of Electronic Systems, 6 (2001), pp. 447–470.

[25] N. Deo and B. Litow, *A structural approach to graph compression*, in Proceedings of MFCS'98 Satellite Workshop on Communications, 1998, pp. 91–101.

[26] H. N. Djidjev and S. M. Venkatesan, *Planarization of graphs embedded on surfaces*, in Proceedings of the 21st Workshop on Graph-Theoretic Concepts in Computer Science, Lecture Notes in Comput. Sci. 1017, Springer, Berlin, 1995, pp. 62–72.

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

[28] A. Farzan and J. I. Munro, *Succinct representations of arbitrary graphs*, in Proceedings of the 16th Annual European Symposium, D. Halperin and K. Mehlhorn, eds., Lecture Notes in Comput. Sci. 5193, Springer, Berlin, 2008, pp. 393–404.

[29] A. Farzan and J. I. Munro, *A uniform approach towards succinct representation of trees*, in Proceedings of the 11th Scandinavian Workshop on Algorithm Theory, Lecture Notes in Comput. Sci. 5124, Springer, Berlin, 2008, pp. 173–184.

[30] A. Farzan, R. Raman, and S. S. Rao, *Universal succinct representations of trees?*, in Proceedings of the 36th International Colloquium on Automata, Languages and Programming, S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. E. Nikoletseas, and W. Thomas, eds., Lecture Notes in Comput. Sci. 5555, Springer, Berlin, 2009, pp. 451–462.

[31] T. Feder and R. Motwani, *Clique partitions, graph compression and speeding-up algorithms*, J. Comput. System Sci., 51 (1995), pp. 261–272.

[32] Y. Feng and D. P. Mehta, *Module relocation to obtain feasible constrained floorplans*, IEEE Trans. CAD Integrated Circ. Syst., 25 (2006), pp. 856–866.

[33] R. Fujimaki and T. Takahashi, *A surjective mapping from permutations to room-to-room floorplans*, IEICE Trans. Fund. Electron. Comm. Comput. Sci., 90-A (2007), pp. 823–828.

[34] H. Galperin and A. Wigderson, *Succinct representations of graphs*, Inform. and Control, 56 (1983), pp. 183–198.

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

[36] C. Gavoille and N. Hanusse, *Compact routing tables for graphs of bounded genus*, in Proceedings of the 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, Berlin, 1999, pp. 351–360.

[37] C. Gavoille and N. Hanusse, *On compact encoding of pagenumber k graphs*, Discrete Math. Theor. Comput. Sci., 10 (2008), pp. 23–34.

[38] C. Gavoille and C. Summer, *Sparse spanners vs. compact routing*, in Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, R. Rajaraman and F. M. auf der Heide, eds., 2011, pp. 225–234.

[39] J. R. Gilbert, J. P. Hutchinson, and R. E. Tarjan, *A separator theorem for graphs of bounded genus*, J. Algorithms, 5 (1984), pp. 391–407.

[40] M. T. Goodrich, *Planar separators and parallel polygon triangulation*, J. Comput. System Sci., 51 (1995), pp. 374–389.

[41] J. L. Gross, *Topological Graph Theory*, Wiley, New York, 1987.

[42] R. Grossi and E. Lodi, *Simple planar graph partition into three forests*, Discrete Appl. Math., 84 (1998), pp. 121–132.

[43] P.-N. Guo, T. Takahashi, C.-K. Cheng, and T. Yoshimura, *Floorplanning using a tree representation*, IEEE Trans. CAD Integrated Circ. Syst., 20 (2001), pp. 281–289.

[44] S. L. Hakimi, *A problem on rectangular floorplans*, in Proceedings of the 1988 IEEE International Symposium on Circuits and Systems, Vol. 2, 1988, pp. 1533–1536.

[45] X. He, *On floor-plan of plane graphs*, SIAM J. Comput., 28 (1999), pp. 2150–2167.

[46] X. He, M.-Y. Kao, and H.-I. Lu, *Linear-time succinct encodings of planar graphs via canonical orderings*, SIAM J. Discrete Math., 12 (1999), pp. 317–325.

[47] X. He, M.-Y. Kao, and H.-I. Lu, *A fast general methodology for information-theoretically optimal encodings of graphs*, SIAM J. Comput., 30 (2000), pp. 838–846.

[48] M. Isenburg and J. Snoeyink, *Spirale Reversi: Reverse decoding of the Edgebreaker encoding*, Comput. Geom., 20 (2001), pp. 39–52.

[49] A. Itai and M. Rodeh, *Representation of graphs*, Acta Inform., 17 (1982), pp. 215–219.

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

[51] Y. Kajitani, *Floorplan and placement*, in Encyclopedia of Algorithms, M.-Y. Kao, ed., Springer, Berlin, 2008, pp. 317–320.

[52] S. Kannan, M. Naor, and S. Rudich, *Implicit representation of graphs*, SIAM J. Discrete Math., 5 (1992), pp. 596–603.

[53] M. Y. Kao, N. Occhiogrosso, and S. H. Teng, *Simple and efficient compression schemes for dense and complement graphs*, J. Comb. Optim., 2 (1999), pp. 351–359.

[54] K.-i. Kawarabayashi, B. Mohar, and B. Reed, *A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width*, in Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, 2008, pp. 771–780.

[55] K. Keeler and J. Westbrook, *Short encodings of planar graphs and maps*, Discrete Appl. Math., 58 (1995), pp. 239–252.

[56] S. Krivograd, M. Trlep, and B. Zalik, *A hexahedral mesh connectivity compression with vertex degrees*, Comput. Aided Des., 40 (2008), pp. 1105–1112.

[57] Y.-T. Lai and S. M. Leinwand, *Algorithms for floorplan design via rectangular dualization*, IEEE Trans. CAD Integrated Circ. Syst., 7 (1988), pp. 1278–1289.

[58] C.-H. Li, S. Sonalkar, and L. P. Carloni, *Exploiting local logic structures to optimize multicore SoC floorplanning*, in Proceedings of the IEEE Conference on Design, Automation and Test in Europe, 2010, pp. 1291–1296.

[59] X. Li, Y. Ma, and X. Hong, *A novel thermal optimization flow using incremental floorplanning for 3D ICs*, in Proceedings of the 14th IEEE Asia South Pacific Design Automation Conference, 2009, pp. 347–352.

[60] Z. Li, X. Hong, Q. Zhou, J. Bian, H. H. Yang, and V. Pitchumani, *Efficient thermal-oriented 3D floorplanning and thermal via planning for two-stacked-die integration*, ACM Trans. Des. Autom. Electron. Syst., 11 (2006), pp. 325–345.

[61] Z. Li, X. Hong, Q. Zhou, Y. Cai, J. Bian, H. H. Yang, V. Pitchumani, and C.-K. Cheng, *Hierarchical 3-D floorplanning algorithm for wirelength optimization*, IEEE Trans. Circuits Syst. I Regul. Pap., 53 (2006), pp. 2637–2646.

[62] J.-M. Lin and Y.-W. Chang, *TCG: A transitive closure graph-based representation for general floorplans*, IEEE Trans. VLSI Syst., 13 (2005), pp. 288–292.

[63] R. J. Lipton and R. E. Tarjan, *A separator theorem for planar graphs*, SIAM J. Appl. Math., 36 (1979), pp. 177–189.

[64] H. Lopes, G. Tavares, J. Rossignac, A. Szymczak, and A. Safanova, *Edgebreaker: A simple compression for surfaces with handles*, in Proceedings of the 7th ACM Symposium on Solid Modeling and Applications, 2002, pp. 289–296.

[65] H.-I. Lu, *Linear-time compression of bounded-genus graphs into information-theoretically optimal number of bits, in Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2002, pp. 223–224.

[66] H.-I. Lu, *Improved compact routing tables for planar networks via orderly spanning trees*, SIAM J. Discrete Math., 23 (2010), pp. 2079–2092.

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

[68] K. Maling, S. H. Mueller, and W. R. Heller, *On finding most optimal rectangular package plans*, in Proceedings of the 19th Conference on Design Automation, 1982, pp. 663–670.
[69] M. D. Moffitt, J. A. Roy, I. L. Markov, and M. E. Pollack, *Constraint-driven floorplan repair*, ACM Trans. Des. Autom. Electron. Syst., 13 (2008), pp. 67.1–67.13.
[70] B. Mohar, *A linear time algorithm for embedding graphs in an arbitrary surface*, SIAM J. Discrete Math., 12 (1999), pp. 6–26.
[71] J. I. Munro and V. Raman, *Succinct representation of balanced parentheses, static trees and planar graphs*, in Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, Miami Beach, FL, 1997, pp. 118–126.
[72] J. I. Munro and V. Raman, *Succinct representation of balanced parentheses and static trees*, SIAM J. Comput., 31 (2001), pp. 762–776.
[73] J. I. Munro, V. Raman, and A. Storm, *Representing dynamic binary trees succinctly*, in Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, 2001, pp. 529–536.
[74] M. Naor, *Succinct representation of general unlabeled graphs*, Discrete Appl. Math., 28 (1990), pp. 303–307.
[75] C. H. Papadimitriou and M. Yannakakis, *A note on succinct representations of graphs*, Inform. and Control, 71 (1986), pp. 181–185.
[76] M. Patrascu, *Succincter*, in Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, 2008, pp. 305–313.
[77] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*, SIAM Monogr. Discrete Math. Appl. 5, SIAM, Philadelphia, 2000.
[78] S. Pettie and V. Ramachandran, *An optimal minimum spanning tree algorithm*, J. ACM, 49 (2002), pp. 16–34.
[79] D. Poulalhon and G. Schaeffer, *Optimal coding and sampling of triangulations*, Algorithmica, 46 (2006), pp. 505–527.
[80] J. Rossignac, *Edgebreaker: Connectivity compression for triangle meshes*, IEEE Trans. Visual. Comput. Graph., 5 (1999), pp. 47–61.
[81] J. Rossignac, A. Safonova, and A. Szymczak, *3D compression made simple: Edgebreaker on a corner-table*, in Proceedings of the International Conference on Shape Modeling and Applications, 2001.
[82] J. Rossignac and A. Szymczak, *Wrap&Zip decompression of the connectivity of triangle meshes compressed with Edgebreaker*, Comput. Geom., 14 (1999), pp. 119–135.
[83] K. Sadakane and G. Navarro, *Fully-functional succinct trees*, in Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, 2010, pp. 134–149.
[84] D. Saha and S. Sur-Kolay, *Encoding of floorplans through deterministic perturbation*, in Proceedings of the 22nd International IEEE Conference on VLSI Design, 2009, pp. 315–320.
[85] J. Snoeyink and M. van Kreveld, *Linear-time reconstruction Delaunay triangulations with applications*, in Proceedings of the 5th Annual European Symposium on Algorithms, Lecture Notes in Comput. Sci. 1284, Springer, Berlin, 1997, pp. 459–471.
[86] S. Sridharan, M. DeBole, G. Sun, Y. Xie, and V. Narayanan, *A criticality-driven microarchitectural three dimensional (3D) floorplanner*, in Proceedings of the 14th IEEE Asia South Pacific Design Automation Conference, 2009, pp. 763–768.
[87] L. J. Stockmeyer, *Optimal orientations of cells in slicing floorplan designs*, Inform. and Control, 57 (1983), pp. 91–101.
[88] T. Suel and J. Yuan, *Compressing the graph structure of the web*, in Proceedings of the 11th IEEE Data Compression Conference, 2001, pp. 213–222.
[89] A. Szymczak, D. King, and J. Rossignac, *An Edgebreaker-based efficient compression scheme for regular meshes*, Comput. Geom., 20 (2001), pp. 53–68.
[90] T. Takahashi, R. Fujimaki, and Y. Inoue, *A $(4n-4)$-bit representation of a rectangular drawing or floorplan*, in Proceedings of the 15th Annual International Conference on Computing and Combinatorics, 2009, pp. 47–55.
[91] K. Tani, S. Tsukiyama, S. Shinoda, and I. Shirakawa, *On area-efficient drawings of rectangular duals for VLSI floor-plan*, Math. Programming, 52 (1991), pp. 29–43.
[92] G. Taubin and J. Rossignac, *Geometric compression through topological surgery*, ACM Trans. Graphics, 17 (1998), pp. 84–115.
[93] C. Thomassen, *The graph genus problem is NP-complete*, J. Algorithms, 10 (1989), pp. 568–576.
[94] M. Thorup, *Compact oracles for reachability and approximate distances in planar digraphs*, J. ACM, 51 (2004), pp. 993–1024.

[95] M. Thorup and U. Zwick, *Compact routing schemes*, in Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architecture, 2001, pp. 1–10.

[96] G. Turán, *On the succinct representation of graphs*, Discrete Appl. Math., 8 (1984), pp. 289–294.

[97] W. T. Tutte, *A census of planar triangulations*, Canad. J. Math., 14 (1962), pp. 21–38.

[98] R. Wang, E. Young, and C.-K. Cheng, *Representing topological structures for* 3*-D floorplanning*, in Proceedings of the International Conference on Communications, Circuits and Systems, 2009, pp. 1098–1102.

[99] R. Wang, E. F. Y. Young, Y. Zhu, F. C. Graham, R. Graham, and C.-K. Cheng, 3*-D floorplanning using labeled tree and dual sequences*, in Proceedings of the 2008 International ACM Symposium on Physical Design, 2008, pp. 54–59.

[100] S. Wimer, I. Koren, and I. Cederbaum, *Floorplans, planar graphs, and layouts*, IEEE Trans. Circuits and Systems, 35 (1988), pp. 267–278.

[101] D. F. Wong, H. W. Leong, and C. L. Liu, *Simulated Annealing for VLSI Design*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1988.

[102] D. F. Wong and C. L. Liu, *Floorplan design of VLSI circuits*, Algorithmica, 4 (1989), pp. 263–291.

[103] K. Yamanaka and S.-I. Nakano, *Coding floorplans with fewer bits*, IEICE Trans. Fund. Electron. Comm. Comput. Sci., E89-A (2006), pp. 1181–1185.

[104] K. Yamanaka and S.-I. Nakano, *A compact encoding of rectangular drawings with efficient query supports*, IEICE Trans. Fund. Electron. Comm. Comput. Sci., 91-A (2008), pp. 2284–2291.

[105] K. Yamanaka and S.-I. Nakano, *A compact encoding of plane triangulations with efficient query supports*, Inform. Process. Lett., 110 (2010), pp. 803–809.

[106] B. Yao, H. Chen, C.-K. Cheng, and R. Graham, *Floorplan representations: Complexity and connections*, ACM Trans. Des. Autom. Electron. Syst., 8 (2003), pp. 55–80.

[107] K.-H. Yeap and M. Sarrafzadeh, *Floor-planning by graph dualization:* 2*-concave rectilinear modules*, SIAM J. Comput., 22 (1993), pp. 500–526.

[108] E. F. Y. Young, *Slicing floorplan orientation*, in Encyclopedia of Algorithms, M.-Y. Kao, ed., Springer, Berlin, 2008, pp. 852–855.

[109] H. A. Zhao, C. Liu, Y. Kajitani, and K. Sakahushi, *EQ-sequences for coding floorplans*, IEICE Trans. Fund. Electron. Comm. Comput. Sci., 87-A (2004), pp. 3244–3250.

[110] C. Zhuang, K. Sakanushi, L. Jin, and Y. Kajitani, *An extended representation of Q-sequence for optimizing channel-adjacency and routing-cost*, in Proceedings of the 2003 Conference on Asia South Pacific Design Automation, 2003, pp. 21–24.