On Compact Encoding of Pagenumber k Graphs

Cyril Gavoille[1] and Nicolas Hanusse[1]

[1] *LaBRI, CNRS, Univ. Bordeaux, 351, cours de la Libération, 33405 Talence, France.* gavoille,hanusse@labri.fr

received Jan. 12, 2001, revised Sep. 22, 2003, Aug. 2, 2004, June 24, 2008, accepted July 21, 2008.

In this paper we show an information-theoretic lower bound of $kn-o(kn)$ on the minimum number of bits to represent an unlabeled simple connected n-node graph of pagenumber k. This has to be compared with the efficient encoding scheme of Munro and Raman of $2kn+2m+o(kn+m)$ bits (m the number of edges), that is $4kn+2n+o(kn)$ bits in the worst-case.

For m-edge graphs of pagenumber k (with multi-edges and loops), we propose a $2m\log_2 k+O(m)$ bits encoding improving the best previous upper bound of Munro and Raman whenever $m \le \frac{1}{2}kn/\log_2 k$. Actually our scheme applies to k-page embedding containing multi-edge and loops. Moreover, with an auxiliary table of $o(m\log k)$ bits, our coding supports (1) the computation of the degree of a node in constant time, (2) adjacency queries with $O(\log k)$ queries of type *rank*, *select* and *match*, that is in $O(\log k \cdot \min\{\log k/\log\log m, \log\log k\})$ time and (3) the access to δ neighbors in $O(\delta)$ runs of *select*, *rank* or *match*.

Keywords: encoding of graphs, k-page embedding, lower bound

1 Introduction

A book is a singular surface composed by the union of closed half-planes, called the *pages*, whose intersection is a line, called the *spine*, being the boundary of each of the pages. A k-page embedding of a graph consists of a linear ordering of the nodes drawn in the spine and of a partition of the edges into k *pages* so that edges residing on the same page do not cross[(i)] (see Fig. 1 for an example). Such an embedding is called *book-embedding*. See [Bil92, DW04] for a survey. Graphs having a k-page embedding are called k-page graphs. The *pagenumber* of a graph G is the smallest integer k such that G is a k-page graph.

The question of the compact coding of a graph (or a part of its topological structure) arises in the field of compact routing [PU89], implicit representation [AR02, KNR92, Lu02], compact representation of 3D-models in computer graphics [KADS02, KR99], distance labeling of graphs [GPPR01] and other label-based informative systems [AKM01, Pel00]. For instance, in the field of compact routing, the goal is to code in the memory of each node s of a graph (representing a communicating network) a routing table that specifies for each destination d the edge to use to route a message from s to d. Pre-computation on

(i) Assuming that $u<v$, $u'<v'$ and $u<u'$, edge (u,v) and edge (u',v') cross whenever $v<v'$.

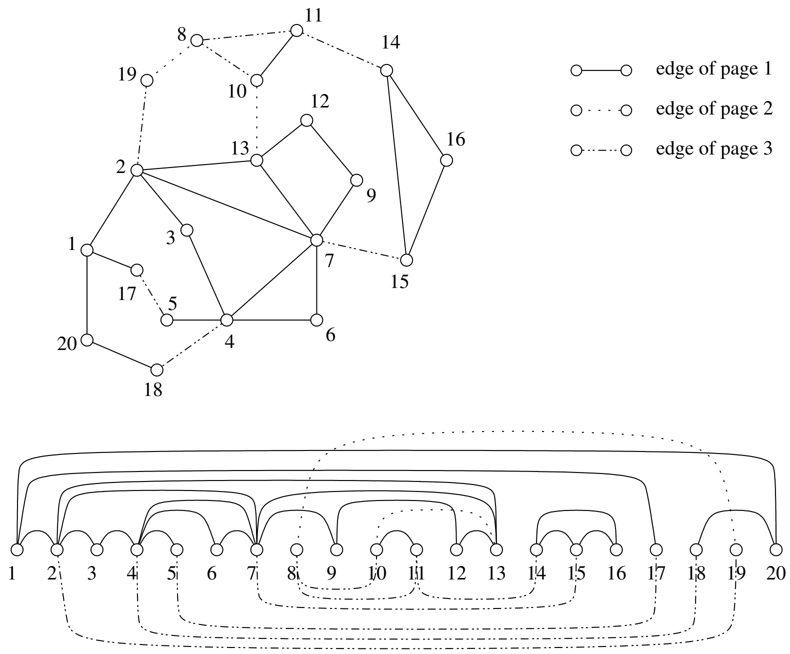


Fig. 1: A planar graph and a 3-page embedding.

the graph structure allows us to reduce the size of routing tables, for instance, assigning suitable address names to the nodes of the graph. (On an n-node ring graph, nodes could be labeled from 1 to n around the ring in order to simplify the routing table data structures and speed up the query time.) For shortest path routing in general n-node graphs, there exists an optimal lower bound of $\Omega(n \log n)$ bits per node [GP96]. However, it as been shown that $O(n \log k)$ bits per node are enough to represent shortest path routing tables for k-page graphs [GH99].

Graphs of pagenumber 1 are the outerplanar graphs (including trees), and graphs of pagenumber 2 are subgraphs of Hamiltonian planar graphs (including series-parallel graphs), cf. [Bil92]. It is NP-hard to compute the pagenumber for general graphs, but there is a linear time algorithm to compute a 4-page embedding of any planar graph. This implies that the pagenumber of every planar graph does not exceed 4 [Yan89]. There are also non-planar graphs with pagenumber $k \leq 4$. For instance K_5 has pagenumber 3.

For arbitrary graphs, there are few algorithms to compute a k-page embedding. Using a probabilistic approach, Malitz [Mal94] proved that the pagenumber of an m-edge graph of *genus* γ is $O(\sqrt{\gamma}\,)$, implying that the pagenumber is $O(\sqrt{m}\,)$ since $\gamma \leq m$. Shahrokhi and Shi [SS00] proposed a deterministic algorithm to embed a node c-colored graph into $O(\sqrt{cm}\,)$ pages whereas Wood [Woo02] described a Las Vegas algorithm to embed a graph of maximal degree Δ into $O(\sqrt{m\Delta/\delta}\,)$ pages such that there are at most δ incident edges per page for every node.

The structure of the paper is the following: Section 2 presents a lower bound of $kn - o(n)$ bits for the representation of graphs with pagenumber k. To our best knowledge, this is the first lower bound on the coding of k-page graphs.

In Section 3 we propose a coding scheme for m-edge k-page embeddings, allowing multiple edges and loops. If the graph has no *isolated nodes*, i.e., no connected components with one node, then the length is at most $2m \log_2 k + 4m$ bits. This improves the Munro and Raman's scheme of $2kn + 2m$ bits [MR01], whenever $m \le \frac{1}{2} kn / \log_2 k$, that is in many pratical cases since a k-page graph has at most $kn + n - 3k$ edges [BK79]. For bounded k and adding an auxiliary table of $o(m)$ bits, our coding supports constant time adjacency queries. For unbounded k, the auxiliary table takes $o(m \log k)$ bits and fast queries can be done: the adjacency test performs in $O(\tau \cdot \log k)$ time, and the degree computation in $O(\tau)$ time, where $\tau = \min\{\log k / \log\log m, \log\log k\}$ are currently the best time complexity for *rank* and *select* operations.

2 Lower Bound

First, we show that $\Omega(kn)$ bits are required in general to code a graph of pagenumber k. On the other hand, Jacobson has shown in [Jac89] that $O(kn)$ bits suffice. So, $\Theta(kn)$ is the optimal length to code a pagenumber k graph on n nodes. For that purpose, we construct a large family of pagenumber k graphs that are pairwise non-isomorphic, and we show that the logarithm of the cardinality of this family is $\Omega(kn)$.

Theorem 1 *The number $\mathcal{U}_{n,k}$ of unlabeled simple connected graphs of n nodes, and pagenumber k, $1 \le k \le n/2$, satisfies*

$$\mathcal{U}_{n,k} \ \ge \ \frac{\left(2^k\left(2^k-1\right)\right)^{n/2-k}}{2\,k!} \ .$$

Proof: Here is a simple proof to show that $\mathcal{U}_{n,k} = 2^{\Theta(kn)}$ for each k. No attempt is made to optimize the constants involved in the calculations. To lower bound $\mathcal{U}_{n,k}$, we construct a family of connected graphs of n nodes and pagenumber k. For $k = 1$, $\mathcal{U}_{n,1}$ is lower bounded by the number of unlabeled trees (which are 1-page graphs [CLR87]) which is given by Otter's formula (cf. [Ric48, Rei05, page 53]):

$$\mathcal{U}_{n,1} \ > \ 0.5349485 \cdot (2.95576)^n \cdot n^{-5/2} \ > \ \frac{2^{n/2-1}}{2 \cdot 1!} \ .$$

Moreover, the formula trivially holds for $n = 2k$, so we assume that $k \ge 2$ and $n > 2k \ge 4$. Note that the pagenumber (as a function of n) is maximal for the complete graph of n nodes, K_n (the family of pagenumber k graphs is hereditary). In this case $k = \lceil n/2 \rceil$, for every $n \ge 4$, and $k = 1$ otherwise (cf. [CLR87]). Let $\mathcal{M}$ be the set of $k \times (n-2k)$ boolean matrices such that every even column has at least one 0-entry. By convenience, this last condition is called the *0-entry condition*. For each matrix $M \in \mathcal{M}$ we define an n-node graph, G_M, as follows: $V(G_M) = \{1, \ldots, n\}$, and $(i,j) \in E(G_M)$ if at least one of the following three cases occur:

1. $|i - j| = 1$;
2. $i, j \le 2k$;
3. $i = 2p - 1, p \in \{1, \ldots, k\}, j > 2k$, and $M_{p,j-2k} = 1$.

$$M = \begin{array}{c} \begin{array}{cccc} 7 & 8 & 9 & 10 \end{array} \\ \left[\begin{array}{cccc} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right] \end{array} \begin{array}{c} \\ 1 \\ 3 \\ 5 \end{array}$$

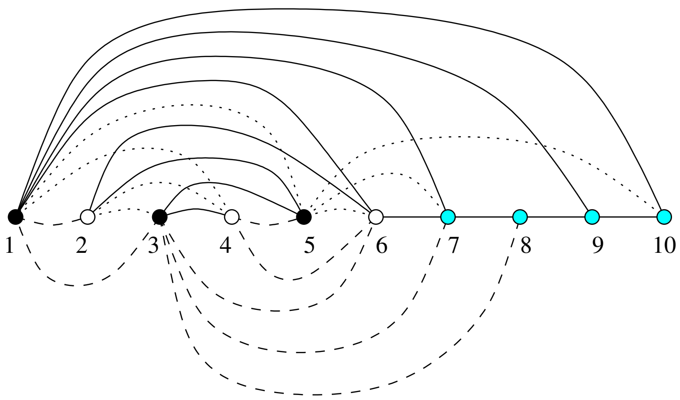


Fig. 2: Construction of an n-node graph of pagenumber $k = 3$. The matrix M denotes adjacency between the nodes $\{1, 3, 5\}$ and $\{7, 8, 9, 10\}$. Solid, dashed and dotted lines correspond respectively to page 1,2 and 3.

See Fig. 2 for an example with $k = 3$ and $n = 10$.

G_M has a Hamiltonian path $(1, 2), \ldots, (i, i + 1), \ldots (n - 1, n)$, therefore G_M is connected. Since for all $i, j \leq 2k$ the edge (i, j) exists, G_M has a clique of size at least $2k$. It follows that its pagenumber is at least the pagenumber of the complete graph K_{2k}, i.e., k since $2k \geq 4$. As we will see later, the 0-entry condition implies the uniqueness of a complete graph K_{2k} in G_M. Let us show that G_M has a k-page embedding, and hence that G_M is connected, has n nodes, and has pagenumber k exactly.

The following lemma deals with a specific embedding of a complete graph K_{2k} proposed in [CLR87]:

Lemma 1 *For every $k \geq 1$, K_{2k} has a k-page embedding such that for every page $p \in \{1, \ldots, k\}$ there is no edge (a, b) embedded on page p with $a < 2p - 1$ and $b > 2p - 1$.*

Proof: Nodes of K_{2k} are labeled from 1 to $2k$. We will assume in the following that the labels of the nodes are given modulo $2k$. We first propose an embedding with $2k - 1$ edges per page and then we prove that no pair of edges belonging to the same page cross. Page $p \in \{1, \ldots, k\}$ contains the set of edges $\{(1 + p + i, 2k - i + p), (2 + p + i, 2k - i + p)\}$ for $i = 0, \ldots, k - 2$ and the edge $(p + k, p + k + 1)$, that is edge (u, v) belongs to page p whenever either $u + v \equiv 2p + 1 \pmod{2k}$ or $u + v \equiv 2p + 2 \pmod{2k}$.

There is no edge (a, b) drawn on page p such that $a < 2p - 1$ and $b > 2p - 1$. Indeed, suppose $a < b \leq 2k$ and $a < 2p - 1$, it follows that $i < p - 2$. For all $i < p - 2$, $2k + p > b = 2k - i + p > 2k + 2$ and b is renamed between 2 and p. This implies that $b < a$ which is in contraction with the hypothesis.

To finish, no pair of edges (u, v) and (u', v') belonging to the same page can cross. Assume without lost of generality that $u < v$, $u' < v'$ and $u < u'$. By definition of the edges embedding, we have $|u + v - u' - v'| \leq 1$. This implies that $v - v' \geq 0$ and $v \geq v'$. $\square$

To embed G_M, first draw the nodes $1, 2, \ldots, n$ on a straight line, and the edges $(2k, 2k + 1), \ldots, (n - 1, n)$ on the first page. All the edges of the clique (point 2 of the definition of G_M) are drawn as in Lemma 1. Then, for each page $p \in \{1, \ldots, k\}$, we draw the edges $(2p - 1, j)$ for every $j \in \{2k + 1, \ldots, n\}$ and $M_{p,j-2k} = 1$. This is possible because these new edges are all incident on the same node, $2p-1$, and by Lemma 1 there is no edge of K_{2k} that spans the node $2p-1$. Hence G_M has a k-page embedding.

Let $\mathcal{G}_{n,k}$ be the set of labeled graphs defined from the set $\mathcal{M}$. Counting the number of ways to fill odd and even columns of matrices of $\mathcal{M}$, we have:

$$\begin{aligned}|\mathcal{G}_{n,k}| = |\mathcal{M}| &= \left(2^k\right)^{\lceil (n-2k)/2 \rceil} \left(2^k-1\right)^{\lfloor (n-2k)/2 \rfloor} \\ &\geq \left(2^k\left(2^k-1\right)\right)^{n/2-k}.\end{aligned}$$

In order to lower bound $\mathcal{U}_{n,k}$, let us upper bound the number of non-isomorphic graphs in $\mathcal{G}_{n,k}$. Let $G \in \mathcal{G}_{n,k}$. Let K be a clique of G of maximal size, i.e., with a maximum number of nodes. Note that, by construction, G has a clique of size $2k$ (the first $2k$ nodes), and also that the size of the maximal clique is at most $2k$ otherwise the pagenumber of G would be at least $\lceil (2k+1)/2 \rceil > k$.

Let us show that K is unique in G, and thus is composed of the nodes labeled $1, \ldots, 2k$. It suffices to show that every node $u > 2k$ belongs to a clique of size at most $k+1$, and thus cannot belong to K, since for every $k \geq 2$, $k+1 < 2k$. Let K_0 be the k odd nodes of $\{1, \ldots, 2k\}$. The j-th column of M corresponds to the node $u = j + 2k$, and defines the connections between u and K_0. Note that there is no edge between $u-1$ and $u+1$, even for $u = 2k+1$. Assume that u is even. In this case, u has degree at most $k+1$ because u neighbors $u-1$, $u+1$, and at most $k-1$ nodes of K_0 since j is even and the 0-entry condition. Thus, u belongs to a clique of size at most $k+2$. However, $u-1$ and $u+1$ are not connected. Therefore, either $u-1$ or $u+1$ does not belong to its clique. So, u belongs to a clique of size at most $k+1$. Now, assume u odd. u is of degree at most $k+2$. u belongs to a clique of size $k+3$ if $u-1$ and $u+1$ both belong to the same clique. If the maximal clique of u contains $u-1$ or $u+1$, its size is at most $k+1$ since $u-1$ and $u+1$ are even, and applying the previous case. Finally, if u contains neither $u-1$ nor $u+1$, the clique is of size at most $k+1$, completing our claim.

$\mathcal{G}_{n,k}$ contains labeled graphs. Remind that two labeled graphs $G = (V, E)$ and $G' = (V, E')$ (defined on the same set of nodes V) are isomorphic if it exists a permutation $g : V \rightarrow V$ of the labels of the nodes such that (u, v) is an edge of G iff the edge $(g(u), g(v))$ belongs to G'. The automorphism group $Aut(G)$ of a graph G is the group of permutations preserving the adjacency, that is the set of isomorphisms of G to itself ($E' = E$). Let $\gamma = \{g\}$ be the set of isomorphisms mapping $G_M \in \mathcal{G}_{n,k}$ to $G_{M'} \in \mathcal{G}_{n,k}$. We have that the number of labelled graphs isomorphic to G_M is equal to $\frac{|\gamma|}{|Aut(G_M)|}$.

The observation of uniqueness of the clique K and the chain $C = 2k+1, \ldots, n$, for any isomorphism g of G, implies that $u \in K \Leftrightarrow g(u) \in K$. More precisely, we have in K: $k+i$ nodes of degree $2k$ (k even nodes and i odd nodes not linked to C) and $k-i$ nodes of degree more than $2k$. Moreover, by definition of the graphs of $\mathcal{G}_{n,k}$, for each node u of the chain C, $g(u) \in \{2k+1, \ldots, n\}$ and $|g(u) - g(u+1)| = 1$. This last property means that there are only two ways of connecting to K (and labeling) the $n-2k$ nodes of C.

Via an isomorphism g, nodes u and $g(u)$ have the same degree, which implies that $|\gamma| \leq 2 \cdot (k+i)!(k-i)!$. Some of the isomorphisms applied to a given graph G_M leads to a graph $G_{M'} = G_M$: the automorphisms.

We trivially have at least $(k+i)!$ automorphisms of the nodes of degree $2k$ in K. So, each graph is isomorphic to at most $2 \cdot (k-i)! \leq 2 \cdot k!$ graphs in $\mathcal{G}_{n,k}$. Therefore there are at least $|\mathcal{G}_{n,k}|/(2k!)$ non-isomorphic graphs in $\mathcal{G}_{n,k}$, completing the proof. □

From the previous counting, $\log_2 \mathcal{U}_{n,k} \geq k(n-2k-O(\log k))$. For every $k = o(n)$, $\log_2 \mathcal{U}_{n,k} \geq kn - o(kn)$. The Munro and Raman's scheme of $2kn + 2m$ bits [MR01], available not only for simple graphs but also for k-page embeddings with multiple edges and loops, implies that $\log_2 \mathcal{U}_{n,k} \leq 4kn + 2n - 6k$ since $m \leq kn + n - 3k$ [BK79].

Corollary 1 *For any $k = o(n)$, every coding scheme of k-page n-node graphs needs at least $kn - o(kn)$ bits for some worst-case graph, which is tight up to a multiplicative factor 4.*

Remark also that for $k = cn$ for c strictly less than $1/2$, the Munro and Raman's scheme is still optimal up to a constant factor depending on c that could be larger than 4.

An interesting combinatorics question would be to tighten the range on the growth constant for k-page graphs,defined by $\sup_k \lim_{n\to\infty} \mathcal{U}_{n,k}{}^{1/(kn)}$, and similarly for k-page embeddings.

Observe that any graph of the family $\mathcal{G}_{n,k}$ introduced in the proof of Theorem 1 has at least $m \geq n - 2k + 2k(2k-1)/2 = n + \Theta(k^2)$ edges. It follows that for each m and n there is an m-edge n-node graph of pagenumber at least $\Omega(\sqrt{m-n}\,)$. So for $m - n = \Omega(m)$, the general $O(\sqrt{m}\,)$ upper bound on the pagenumber is tight.

3 An Encoding Scheme

The coding of k-page graphs of Jacobson [Jac89] in $9kn$ bits shows that the lower bound of Corollary 1 is asymptotically tight up to a multiplicative constant. Munro and Raman in [MR01] improved this bound to $2kn+2m+o(kn+m)$ bits, where m is the number of edges, supporting $O(k)$ time adjacency queries (in a word-RAM model with $O(\log m+\log n)$-bit words[(ii)]). This coding is always less than $4kn+2n+o(kn)$ bits since the maximum number of edges of a k-page graph is $n(k+1) - 3k$ [BK79]. We can also use the following $2m\log_2 k + 4m$ bit encoding which improves the previous one whenever $m \leq \frac{1}{2}kn/\log_2 k$ and the graph does not contain isolated nodes.

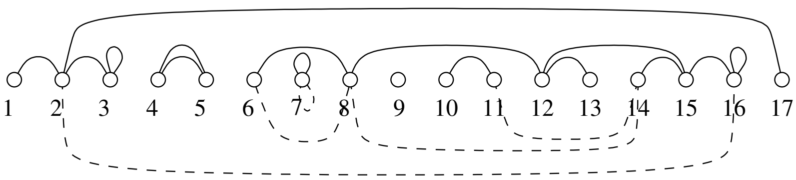
In the following, S and B are respectively a string defined on the alphabet $\{\,(_1,)_1, (_2,)_2, \ldots, (_k,)_k\}$ and a bitmap. For fixed p, the substring of parentheses composed of parentheses of type $(_p,)_p$ is *balanced* if it contains the same number of open and closed parentheses and such that any prefix of the string contains a majority (or equality) of open parentheses. W.l.o.g. a string of multiple parentheses is balanced if all of its subtring of type p is balanced.

To provide quickly adjacency queries, we use the following operations:

- Let *select*(u) be the position of the u-th occurrence of 1 in B and *select*$_p(i,b)$ be the i-th position of the parenthesis $b \in \{\,(_p,)_p\}$ in S.
- Let *rank*(i) be the number of 1 in B before and including position i and *rank*$_p(i,b)$ be the number of parenthesis $b \in \{\,(_p,)_p\}$ in S before (and including) the position i.
- Let *match*(i) be the position in S of the matching parenthesis of $S[i]$.

The first efficient implementation of balanced string of k multiple types of parentheses and bitmaps for several queries have been proposed in [CGH+98, CLL01, MR01] whenever $k = O(1)$. For a string of length ℓ and $2k$ symbols, it follows that adding an auxiliary table of $o(\ell \log k)$ bits, operations *select* and *rank* can can be done in $O(\log\log k)$ time [GMR06], or alternatively in $O(\log k/\log\log \ell)$ time [FMMN07], which is constant for $k = O(\text{polylog}(\ell))$.

(ii) We will consider later multi-edge graphs, so m may be not bounded above y a function of n, $m > 2^n$ is possible.



S = ( )[(( )() (( )) ([([)])]([() ( )[ )(( ) (]]))( )]())

B = 1 1000 100 10 10 10 1000 1000 10 1 10 100 1 100 100 1000 1

Fig. 3: A 2-page embedding and its coding (S, B).

Theorem 2 *Every k-page embedding with m edges (possibly with multiple edges and loops) and i isolated nodes can be represented with $2(m+i)\log_2 k + 4(m+i)$ bits. Moreover, it is possible (1) to check adjacency with at most $O(\log k)$ runs of operations* select, rank *or* match, *(2) to compute the degree of a node with a constant number runs of operations* select ,rank *and (3) to list the δ neighbors of a node in $O(\delta)$ runs of* select, rank *or* match.

Proof: We consider a given k-page embedding with m edges.

Coding To describe the embedding we use two strings:

- a balanced string S of multiple types of parentheses. Each edge is encoded by a pair of parentheses of a given type; and
- a bitmap B by which the nodes can be located in S.

The use of a string of multiple parentheses is inspired from [CGH+98]. We first assume that the graph does not contain any degree-0 node (i.e., isolated node without loops), then we give an extension of the coding whenever the embedding contains some. The string S is defined as follows (see Fig. 3 for an example): successively for each node $u = 1, \ldots, n$, we assign a block S_u composed of three sequences of closed and open parentheses denoted by $S^{)}_u$ and $S^{(}_u$ and S^{l}_u merged in the order $S^{)}_u$, $S^{(}_u$, S^{l}_u.

1. To built $S^{)}_u$: for each page $p \in \{1, \ldots, k\}$, we encode contiguously one symbol $)_p$ per edge that belongs to page p and that connects a node $v < u$; and we denote by $\Gamma^{)}_p(u)$ the smallest v. Within $S^{)}_u$, the order of the blocks of closed parentheses is the increasing order of $\Gamma^{)}_p(u)$'s.
2. To built $S^{(}_u$: for each page $p \in \{1, \ldots, k\}$, we encode contiguously one symbol $(_p$ per edge that belongs to the page p and that connects a node $v \geq u$; and we denote by $\Gamma^{(}_p(u)$ the largest v. Within $S^{(}_u$, the order of the blocks of open parentheses is the increasing order of $\Gamma^{(}_p(u)$'s.
3. Finally, S^{l}_u encodes a possible loop by adding a symbol $)_p$ for each loop of u on the page p.

The resulting string S has length $2m$ and is composed of $2k$ distinct symbols. Since the drawing of the edges on each page p is outerplanar, the string composed of the symbols $(_p$ and $)_p$ consists of a balanced string of parentheses.

The nodes can be located in S by an extra bitmap B of $2m$ bits. We associate with each parenthesis a bit in the same order than of S. The bit 1 represents the first parenthesis of a node, i.e., the beginning of block S_u in S, and the bit 0 represents the others. So, in total we use $2m \log_2(2k) + 2m = 2m \log_2 k + 4m$ bits.

To recover the embedding from the string S and the bitmap B, we determine for each node u the substring S_u of S associated with u and computed as follows: find positions p_0 and p_1 of the u-th and the $(u+1)$-th bit 1 in B, and extract the substring of S between positions p_0 and $p_1 - 1$ (included).

Two nodes $u \leq v$ are connected by an edge embedded on page p if and only if S_u contains symbol $(_p$, S_v contains symbol $)_p$, and the two typed parentheses match in S [(iii)].

However, we remark that nodes without any incident edges (i.e., isolated nodes without loops) are not encoded by this scheme. To overcome this problem we add an extra loop embedded on page 1 at all isolated nodes (including those with loops in the original embedding), and we code this new embedding as presented previously so that every node has non-null degree. The number of edges increases to $m + i$, where i is the total number of isolated nodes, and so the resulting coding uses $2(m+i)\log_2 k + 4(m+i)$ bits. To recover the original embedding, we extract S_u as before and first check whether u is isolated or not. It is isolated if and only if it contains only pairs of matching parentheses. If it is the case, we remove from S_u one copy of matching symbols $(_1$ and $)_1$, and perform adjacency as previously.

Queries The simplest query is the computation of the degree of a node u: it corresponds to the difference of the $(u+1)$-th position of 1 and the u-th position of 1 in bitmap B, that is $\mathit{select}(u+1) - \mathit{select}(u)$.

In the following, we prove that the adjacency can be done quickly.

We shall first prove that for a given page, the adjacency test between two nodes $u \leq v$ on a given page takes at most a constant number of runs of operations *rank*, *select* and *match*. Then, we shall see that adjacency in the graph can be done by checking adjacency on at most $O(\log k)$ pages.

Let us first observe that, if $u \leq v$ are adjacent in page p then, $|S_u^{(}|$ and $|S_v^{)}|$ are both non-null, and thus $\Gamma_p^{(}(u)$ and $\Gamma_p^{)}(v)$ exist. In that case, and due to the outerplanarity of page p, either $\Gamma_p^{(}(u) = v$ or $\Gamma_p^{)}(v) = u$.

To know whether the nodes $u \leq v$ are adjacent or not in page p, we therefore apply the following steps:

1. Find the first open parenthesis of type p belonging to S_u.
   Return NO if such parenthesis does not exist.

2. Search its matching parenthesis and its corresponding node $\Gamma_p^{(}(u)$.

3. If $\Gamma_p^{(}(u) = v$, then return YES.

4. Find the last closed parenthesis of type p belonging to S_v.
   Return NO if such parenthesis does not exist.

5. Search its matching parenthesis and its corresponding node $\Gamma_p^{)}(v)$.

6. If $\Gamma_p^{)}(v) = u$, then return YES.

[(iii)] A closed parenthesis) matches an open parenthesis (in position i if it is the first closed parentheses in position $j > i$ for which the difference of the number of open and the number of closed parenthesis between them is null.

Step 1 can be done using *select* in B to find blocks S_u in S and its starting position $s_u = \textit{select}(u)$. Let l_u be the position in S of the first open parenthesis of type p in S_u. To get l_u, we will count the number of $(_p$'s before s_u and select the next occurrence of $(_p$. It follows that $l_u = \textit{select}_p(\textit{rank}_p(s_u-1, (_p)+1, (_p)$. Comparing the position $\textit{select}(u+1)$, which is equal to $s_u + |S_u|$, and the position l_u, we can detect if $|S_u^(| > 0$ or not, and possibly answer NO to the adjacency test.

The matching node of the parenthesis in position l_u in S is precisely $\Gamma_p^((u)$, which can be computed by: $\Gamma_p^((u) = \textit{rank}(match_p(l_u, (_p))$, the *rank* operation being computed in B. This completes Step 2, Step 3 is trivial.

Steps 4-6 are quite similar: we should distinguish two subcases: (1) either v has a loop in page p or (2) v has no loop in page p. This can be done by determining the position r_v in S of the matching parenthesis of the last closed parenthesis of type p: $r_v = \textit{match}_p(\textit{rank}(s_{v+1} - 1,)_p))$. If $r_v \geq s_v$, then v has a loop. If $u \neq v$, we need to do an extra work to find the last closed parentheses of $S_v^)$ of type p: $r = \textit{select}_p(rank_p(r_v,)_p))$. Then $\Gamma_p^)(v) = \textit{rank}(match_p(r,)_p))$. If $\Gamma_p^)(v) \neq u$ then there is no edge (u, v) in page p.

Obviously, some computation may failed, if $|S_v^(| = 0$ for instance, which can be easily detected. For $u = v$, we must return NO for the adjacency test.

Therefore, the adjacency test in a given page takes a constant number of runs of operations *select*, *rank*, and *match*.

To check the existence of an edge between $u \leq v$ in the graph, there is no need to check the adjacency successively in the k pages but in $\lceil \log k \rceil$ pages only.

Remind that block $S_u^(= O_{p_1} O_{p_2} \ldots O_{p_{k_u}}$, where each $O_{p_i} = (_{p_i} (_{p_i} \ldots (_{p_i}$ is a block of open parenthesis on page p_i. By construction of $S_u^($, the sub-blocks O_{p_i}'s have been ordered such that $\Gamma_{p_1}^((u) \leq \Gamma_{p_2}^((u) \leq \cdots \leq \Gamma_{p_{k_u}}^((u)$. Clearly, if there is some page p such that $\Gamma_p^((u) = v$, then $v \in [\Gamma_{p_1}^((u), \Gamma_{p_{k_u}}^((u)]$. A binary search of v in the ordered sequence of $\Gamma_{p_i}^((u)$'s will find such a p (if any) in repeating at most $\lceil \log k_u \rceil$ time steps 1-3.

Similarly, $S_v^) = C_{q_1} C_{q_2} \ldots C_{q_{k_v}}$, where $C_{p_i} =)_{p_i})_{p_i} \ldots)_{p_i}$, and if there is some page q such that $\Gamma_q^)(v) = u$, then $u \in [\Gamma_{q_1}^)(v), \Gamma_{q_{k_v}}^)(v)]$. Such q can be determined (if any) in repeating at most $\lceil \log k_v \rceil$ time steps 4-6.

Overall, the adjacency test can be performed in at most $2 \lceil \log k \rceil$ adjacency test on a given page.

From the above description, listing the neighbors of u is done by reading the block S_u (from position $\textit{select}(u)$), computing the set of matching parenthesis and determining the corresponding nodes (using *rank* in B).

That completes the proof. □

Let k' be the number of non empty pages. Although our coding works with empty pages, in the spirit of Theorem 2, $k' = k$ pages are assumed to be non empty. If the majority of the pages are empty, a bitmap of size k can code this information and an index table of size $o(k)$ can be used to tell in constant time whether a given page is empty. In this case, our coding can be adapted to $2(m+i)\log_2 k' + 4(m+i) + k$ bits.

In Theorem 2, we can use the best description of *rank*, *select* and *match* in the literature. As described in the beginning of this section, recent papers [GMR06, FMMN07] provide description of *rank* and *select* in $O(\min\{\log k/\log\log m, \log\log k\})$ time adding extra index tables of $o(m \log k)$ bits.

Jacobson or Geary *et al.* works [Jac89, CGH+98, GRRR06] show how to use *rank* and *select* to built operation *match*. More precisely, we can assign an auxiliary table of $o(\ell)$ bits to a balanced string of parentheses of length ℓ to get the matching parenthesis of any parenthesis using a constant number of runs of *rank* and *select* operations. Assume that page p has m_p edges. A careful reading of [Jac89, CGH+98, GRRR06] shows that k auxiliary tables of size $o(m_p)$ is enough to run operation *match* in a (multiple type) string S of length [(iv)] $2m$. However, their description is provided for $k = 1$ and is only relevant whenever $k = O(1)$. The following lemma extends their result.

Lemma 2 *Let S be a balanced string of k types of parentheses of length $2m$. Adding an extra bitmap T of $4m + o(m)$ bits, operation* match *in S can be done using a constant number of runs of operations* select, rank *in S and T and one run of* match *in T.*

Proof: Roughly speaking, we shall add to our data structure k extra strings of balanced parenthesis (of single type) $T_1, T_2, \ldots, T_k$ such that T_p corresponds to the subword of S of type p. Operation $j = \mathit{match}_p(i)$ (for $(_p$ in position i) in S can be done using the following sequence:

- find the rank of the current parenthesis (of type p) in S and select the corresponding parenthesis in T_p: $s = \mathit{select}_{T_p}(\mathit{rank}_p(i, (_p), ()$;
- compute the rank of the matching parenthesis in T_p: $r = \mathit{rank}_{T_p}(\mathit{match}_{T_p}(s),))$;
- select the corresponding position in S: $j = \mathit{select}_p(r,)_p)$.

Keep in mind that each T_p is of single type. If each T_p is considered separately, adding index tables of $O(m_p \log m_p / \log\log m_p)$[RRR02] is enough to run operations rank_{T_p}, select_{T_p} and match_{T_p} in constant time.

In order to get the desired data structure, we need to access to the i-th table or index table. To limit the space consumption, we assign index tables to T_p if and only if $m_p > 1/2 \log_2 m$. In the other case, the answer will be precomputed in another table.

To sum up, to represent T_p's and the index tables, we built a bitmap T composed of: (1) the concatenation of $T_1, \ldots, T_k$; (2) three tables that allows the three operations on T_p in $O(1)$ time whenever $m_p \leq \frac{\log_2 m}{2}$ (by precomputing the answers stored within $O(\log\log m)$ bits for any input - a small block and a integer less than $(\log_2 m)/2$); (3) we assign to each T_p auxiliary index tables of size $o(m_p)$ bits as soon as $m_p > \frac{\log m}{2}$; (4) we add to T a field separator bitmap of length $2m + o(m)$ and an index table of size $o(|T|)$ identifying the position of the ith table (among at most $2k$ tables for part 1 and 3 of T) using operation *select*.

We now calculate the space usage. The four parts of our data structure take respectively $2m$ bits, $O(2^{\log_2 m/2} \log m \log\log m) = o(m)$ bits, $\sum_{m_p \geq (\log_2 m)/2} O(m_p \log m_p / \log\log m_p) =$
$O(\frac{\log\log m}{\log\log\log m} \sum_{m_p} m_p) = o(m)$ and $2m + o(m)$. □

(iv) m includes here extra edges for coding isolated notes.

From Theorem 2 and Lemma 2, we get:

Theorem 3 *Every k-page embedding with m edges, with multiple edges and loops, and i isolated nodes can be represented with $2(m+i)\log_2 k + 4(m+i) + o((m+i)\log k)$ bits. Moreover, it is possible to test the adjacency in $O(\tau \cdot \log k)$ time, where $\tau = \min\{\log k/\log\log m, \log\log k\})$. The computation of the degree can be done in $O(1)$ time and listing the δ neighbors of a node takes $O(\tau\delta)$.*

4 Conclusion

In this paper, we focus on representing a k-page embedding. We observe that our coding can be shortened if the task is to represent a simple graphs (no loops and no multi-edges), rather than an embedding. In particular, the additive term $4(m+i)$ can be reduced to $2m+2n$ where n is number of nodes removing bitmap B and coding each node on the first page by a pair of open and closed parenthesis.

In this article, we do not focus on the possibility of compressing strings that takes into account the empirical entropy of the strings. The work of Ferragina *et al.* [FMMN07] can be used to compress our data structure.

It would be interesting to get the adjacency test and the list of neighbors in constant time for arbitrary k. To do this, basic operations *rank* and *select* should take constant time. However, to our knowledge, this problem is open.

Acknowledgments: The authors would like to thank the anonymous referees who made many interesting remarks.

References

[AKM01] Serge Abiteboul, Haim Kaplan, and Tova Milo. Compact labeling schemes for ancestor queries. In 12^{th} *Symposium on Discrete Algorithms (SODA)*, pages 547–556. ACM-SIAM, January 2001.

[AR02] Stephen Alstrup and Theis Rauhe. Small induced-universal graphs and compact implicit graph representations. In 43^{rd} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 53–62. IEEE Computer Society Press, November 2002.

[Bil92] Tomasz Bilski. Embedding graphs in books: A survey. *IEE Proceedings-E*, 139(2):134–138, March 1992.

[BK79] Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27:320–331, 1979.

[CGH+98] Richie Chih-Nan Chuang, Ashim Garg, Xin He, Ming-Yang Kao, and Hsueh-I Lu. Compact encodings of planar graphs via canonical orderings and multiple parentheses. In 25^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1443 of Lecture Notes in Computer Science, pages 1–12. Springer, July 1998.

[CLL01] Yi-Ting Chiang, , Ching-Chi Lin, and Hsueh-I Lu. Orderly spanning trees with applications to graph encoding and graph drawing. In 12^{th} *Symposium on Discrete Algorithms (SODA)*, pages 506–515. ACM-SIAM, January 2001.

[CLR87] Fan R.K. Chung, Frank Thomson Leighton, and Arnold L. Rosenberg. Embedding graph in books: A layout problem with applications to VLSI design. *SIAM Journal on Algebraic Discrete Methods*, 8:33–58, 1987.

[DW04] Vida Dujmović and David R. Wood. On linear layouts of graphs. *Discrete Mathematics and Theoretical Computer Science*, 6(2):339–358, 2004.

[FMMN07] Paolo Ferragina, Giovanni Manzini, Veli Mäkinen, and Gonzalo Navarro. Compressed representations of sequences and full-text indexes. *ACM Transactions on Algorithms*, 3(2):A20, May 2007.

[GH99] Cyril Gavoille and Nicolas Hanusse. Compact routing tables for graphs of bounded genus. In 26^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1644 of Lecture Notes in Computer Science, pages 351–360. Springer, July 1999.

[GMR06] Alexander Golynski, J. Ian Munro, and S. Srinivasa Rao. Rank/select operations on large alphabets: a tool for text indexing. In *SODA*, pages 368–373. ACM Press, 2006.

[GP96] Cyril Gavoille and Stéphane Pérennès. Memory requirement for routing in distributed networks. In 15^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 125–133. ACM Press, May 1996.

[GPPR01] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. In 12^{th} *Symposium on Discrete Algorithms (SODA)*, pages 210–219. ACM-SIAM, January 2001.

[GRRR06] Richard F. Geary, Naila Rahman, Rajeev Raman, and Venkatesh Raman. A simple optimal representation for balanced parentheses. *Theor. Comput. Sci.*, 368(3):231–246, 2006.

[Jac89] Guy Jacobson. Space-efficient static trees and graphs. In 30^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 549–554. IEEE Computer Society Press, October 1989.

[KADS02] Andrei Khodakovsky, Pierre Alliez, Mathieu Desbrun, and Peter Schröder. Near-optimal connectivity encoding of 2-manifold polygon meshes. *Graphical Models*, 2002. To appear in a special issue.

[KNR92] Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. *SIAM Journal on Discrete Mathematics*, 5:596–603, 1992.

[KR99] Davis King and Jarek Rossignac. Guaranteed 3.67V bit encoding of planar triangle graphs. In 11^{th} *Canadian Conference on Computational Geometry (CCCG)*, pages 146–149, August 1999.

[Lu02] Hsueh-I Lu. Linear-time compression of bounded-genus graphs into information-theoretically optimal number of bits. In 13^{th} *Symposium on Discrete Algorithms (SODA)*, pages 223–224. ACM-SIAM, January 2002.

[Mal94] Seth M. Malitz. Genus γ graphs have pagenumber $O(\sqrt{\gamma})$. *Journal of Algorithms*, 17(1):85–109, 1994.

[MR01] J. Ian Munro and Venkatesh Raman. Succinct representation of balanced parentheses, static trees and planar graphs. *SIAM Journal on Computing*, 31(3):762–776, 2001.

[Pel00] David Peleg. Informative labeling schemes for graphs. In 25^{th} *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 1893 of Lecture Notes in Computer Science, pages 579–588. Springer, August 2000.

[PU89] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, 36(3):510–530, July 1989.

[Rei05] Diestel Reinhard. *Graph Theory (3rd ed.)*. Springer Verlag, 2005.

[Ric48] Otter Richard. The number of trees. *Annals of Mathematics*, 49(3):583–599, 1948.

[RRR02] R. Raman, V. Raman, and S.S. Rao. Succinct representations of lcp information and improvments in the compressed suffix arrays. In *ACM-SIAM SODA*, pages 233–242, 2002.

[SS00] Farhad Shahrokhi and Weiping Shi. On crossing sets, disjoints sets, and pagenumber. *Journal of Algorithms*, 34(1):40–53, 2000.

[Woo02] David R. Wood. Degree constrained book embeddings. *Journal of Algorithms*, 45(2):144–154, 2002.

[Yan89] Mihalis Yannakakis. Embedding planar graphs in four pages. *Journal of Computer and System Sciences*, 38:36–67, 1989.