Optimal Succinct Representations of Planar Maps*

Luca Castelli Aleardi† Olivier Devillers‡ Gilles Schaeffer§

Published in Proc. 22th Annu. ACM Sympos. Comput. Geom., 309–318,2006

Abstract

This paper addresses the problem of representing the connectivity information of geometric objects using as little memory as possible. As opposed to raw compression issues, the focus is here on designing data structures that preserve the possibility of answering incidence queries in constant time. We propose in particular the first optimal representations for 3-connected planar graphs and triangulations, which are the most standard classes of graphs underlying meshes with spherical topology. Optimal means that these representations asymptotically match the respective entropy of the two classes, namely 2 bits per edge for 3-connected planar graphs, and 1.62 bits per triangle or equivalently 3.24 bits per vertex for triangulations.

1 Introduction

1.1 Connectivity vs geometry

A geometric object is often represented by a polygonal mesh which contains two kinds of information: the geometry and the connectivity. The connectivity is a graph which describes how vertices are linked by edges and faces, while the geometry consists in the vertices coordinates. In usual representations such as VRML format or pointers representations in main memory [5], the connectivity is the most expensive part: it costs hundreds of bits per vertex while the geometry costs only tens of bits per vertex. As a matter of fact, in such formats where the connectivity is represented by numbering the vertices and giving indexes, the cost of connectivity has order $\Theta(n \lg n)$. Since this is the most expensive part, we shall concentrate in this paper on reducing the connectivity cost, and leave the problem of reducing the geometry cost aside. We observe however that our structure is compatible with several of the standard approaches to geometry compression (point coordinates can be given in a local framework).

*This work has been supported by the French "ACI Masses de données" program, via the Geocomp project, `http://www.lix.polytechnique.fr/~schaeffe/GeoComp/`

†LIX - INRIA Sophia Ecole Polytechnique Palaiseau, France `amturing@lix.polytechnique.fr`

‡INRIA Sophia Antipolis 2004, route des Lucioles Sophia-Antipolis, France `olivier.devillers@sophia.inria.fr`

§LIX Ecole Polytechnique Palaiseau, France `schaeffe@lix.polytechnique.fr`

1.2 Compression vs succinct structures

A geometric object can be represented either in a linear format for disk storage or network transmission, or stored in main memory for the exploration of the object. In the first case, reducing the size is called *compression*, and compressing the connectivity of various kind of meshes has been successfully attacked in recent years [17, 18, 12, 20, 19, 8, 13] (for a survey of the most recent advances in this field we refer to [3]). In this paper we deal with the second case of main memory representation, and our aim is to design a data structure having a small size and allowing to answer queries in constant time. Usual queries consist in going from a face to its neighbor or asking if two vertices are adjacent in the mesh. Such a structure is called *succinct* if the cost asymptotically matches the entropy of the class and *compact* if it matches it up to a constant factor. More precisely, given a class of objects with a size parameter n (e.g. the number of elements of some kind), we consider the number of objects of size n in the class. If this number has an exponential growth of order $2^{\alpha n}$ when n goes to infinity, the *entropy* of the class is defined to be αn. A representation is then *compact* if it uses $O(n)$ bits and *succinct* if it uses $\alpha n + o(n)$ bits. Observe that a correct representation cannot use $o(\alpha n)$ bits for it must be possible to distinguish all objects.

1.3 General framework

The general framework we use for designing a compact or succinct data structure for a given class of objects of size n is sketched here:

- First the object is split into tiny pieces of size $O(\lg n)$, *tiny* meaning small enough so that a catalog of all possible pieces can be constructed in $o(n)$ time and space. Then a tiny piece is represented by its index in the catalog, and the sum of the sizes of all indexes is expected to match the entropy of the class.

- The incidence relations describing how the splitting into tiny pieces has been done is encoded in a graph $\mathcal{G}$ of tiny pieces. Since there are $O(\frac{n}{\lg n})$ tiny pieces with a linear number of incidences between them, a classical representation of this graph using pointers of logarithmic size costs $O(n)$ and this approach already yields a compact data structure.

- *Small* pieces of $O(\lg^2 n)$ size are constructed by joining $\lg n$ tiny pieces, this allows to use pointers of size $O(\lg n)$ only between small pieces while adjacencies between tiny pieces are described with local pointers of size $O(\lg \lg n)$. Since the number of small and tiny pieces are respectively $O(\frac{n}{\lg^2 n})$ and $O(\frac{n}{\lg n})$, this multi-level approach yields sublinear costs of $O(\frac{n \lg n}{\lg^2 n})$ and $O(\frac{n \lg \lg n}{\lg n})$ for $\mathcal{G}$, making the structure succinct.

1.4 Related results

We briefly review in this section a few results about representations of graph connectivity for 3-connected planar graphs and triangulations. These results can be given in terms of n the number of vertices, m the number of faces or e the number of edges.

In the special case of triangulations of a topological sphere, $6n \simeq 2e = 3m$ and the entropy is $1.62m = 3.24n$ bits [21]. For planar triangulations with a boundary, the entropy is $2.175m$ bits. For general 3-connected planar graphs the entropy is $2e$ bits [22].

On the practical side, classical main memory representations use pointers: $n + 6m$ pointers are needed for triangulations and $n + 6e$ for 3-connected graphs [5], where a pointer means 32 bits with real pointers and $\lg n$ bits using indexes. A cheaper solution with $2e$ pointers [14] has been proposed with the price of a higher access cost to neighbors. None of these $O(n \lg n)$ structures are compact.

The above framework has been introduced for balanced parenthesis words (Dyck words) by Jacobson [11] for the compact representation, and by Munro and Raman [15] for the succinct representation.

The size parameter of a parenthesis word is its number of characters and optimality means 1 bit per character. In this context the natural query is to ask for the matching parenthesis of the parenthesis at a given position. These results on parenthesis words allow for succinct representations of plane (aka ordered) trees using $2n$ bits. Then a planar map can be decomposed into several trees which can be succinctly represented.

However, this transformation from graphs to trees being non bijective, it yields representations for planar graphs that are not succinct but only compact. Along these lines, a representation of planar graphs using $2e + 8n$ bits was given [15] and then improved to $2e + 2n$ bits [7, 6].

In our previous work [1], we have shown how to extend the framework so that it can be applied directly to triangulations with a boundary and provided a succinct representation for this (larger) class of triangulations. This approach was also successful in dealing with local dynamic updates of triangulations of higher genus surfaces [2].

With a slightly different strategy, Blandford *et al.* [4] showed how to design a compact data structure supporting adjacency and degrees queries on vertices for special classes of graphs having small separators. However this approach needs efficient algorithms for finding separators and the exact cost of the representation is difficult to characterize. As in previous works, our model of computation is a RAM machine, with $O(1)$ time access on words of size $\Theta(\lg n)$.

1.5 Contribution

The contribution of this paper is twofold. As far as results are concerned, we propose the first succinct data structures for representing planar triangulations without boundary (triangulations of a topological sphere) and 3-connected planar maps, as stated in Theorem 1 below.

Theorem 1 *There exist succinct representations*

- *of planar triangulations (without boundaries) requiring asymptotically* 1.62 *bits per triangle,*
- *of* 3*-connected planar graphs requiring asymptotically* 2 *bits per edge.*

Both representations support local standard navigation in $O(1)$ *time, using an extra storage of size* $O(\frac{n \lg \lg n}{\lg n})$.

From the methodological point of view, we formalize the catalog-tiny-small framework. With respect to the seminal data structures proposed for words and trees [11, 15], and for triangulations in our paper [1], the present framework makes explicit the local planarity properties on which the approach is based.

Furthermore it relaxes the property, central in those previous works, that tiny pieces should be taken in a class with the same entropy as the main class of objects represented. Finally, in order to develop our two new applications (triangulations and 3-connected planar maps), we design new splitting schemes.

Next section will formalize the catalog-tiny-small framework, while Section 3 describes its use for planar maps and Section 4 for triangulations.

2 The catalog-tiny-small framework

In this section we make gradually more precise the general framework sketched in Section 1. At a first level of details, the framework applies to any data structure which has linear entropy and can be decomposed into regions connected by a globally sparse graph: this includes parenthesis words, trees, and graphs on surfaces. The framework is then specialized to connectivity structures of meshes.

2.1 Additivity of the entropy and the catalog

In order for the framework to apply to a class of combinatorial objects, we first need that the entropy be linear. More precisely consider a class $\mathcal{C} = (\mathcal{C}_n)$ of objects where n is intended as a size parameter (the number of some elementary *cells*) and assume that the set $\mathcal{C}_n$ of objects of size n has a finite cardinality $|\mathcal{C}_n|$. The entropy $||\mathcal{C}_n||$, defined by

$$||\mathcal{C}_n|| := \lg |\mathcal{C}_n|$$

with $\lg(x) = \lceil \log_2(x+1) \rceil$, measures the diversity of the class. A class has linear entropy if there exists a constant α such that

$$||\mathcal{C}_n|| = \alpha n + o(n), \quad \text{when } n \text{ goes to infinity.}$$

In other terms, the cardinality of the class $\mathcal{C}_n$ grows roughly like a simple exponential $2^{\alpha n}$, and αn bits are needed to index all elements. The constant α is sometimes called the entropy per size unit: $\alpha = 2$ bits per node for binary trees, and, as indicated above, $\alpha = 1.62$ bit per triangles for triangulations, and $\alpha = 2$ bits per edge for 3-connected planar graphs. Observe however that some classes, like the classes of permutations of $\{1, \ldots, n\}$ or of general n-vertex graphs have non linear entropies, of order $n \lg n$.

We intend to decompose each object of $\mathcal{C}$ into pieces taken from a catalog of smaller objects: as opposed to previous works, we do not require that this catalog contains elements of $\mathcal{C}$, but rather that it contains elements of a class $\mathcal{D} = (\mathcal{D}_{m,k})$, such that there exists a constant β and a positive function $g(m) = O(\lg m)$ such that

$$||\mathcal{D}_{m,k}|| \leq \alpha m + \beta \; k \lg m + g(m) \tag{1}$$

and $|\mathcal{D}_{m,k}| = 0$ if $k \geq Km$ (for some constant K).

The objects of $\mathcal{D}_{m,k}$ are intended to be used to describe tiny pieces of elements of $\mathcal{C}$, with m the number of elementary cells, and k a parameter, called the *number of sides*, describing the complexity of the boundary of the piece. In the above bound the constant α is expected to be the same as for $\mathcal{C}_n$, and αm is the *additive part* of the entropy, while $\beta k \lg m$ is the extra amount of entropy due to the splitting into tiny pieces. By definition of the entropy, $||\mathcal{D}_{m,k}||$ bits are sufficient to index an element of $\mathcal{D}_{m,k}$ in a table representing all those elements.

We assume more precisely that each element M of $\mathcal{C}_n$ can be decomposed into

- a collection $(M_1, \ldots, M_p)$ of elements of $\mathcal{D}$, with $M_j \in \mathcal{D}_{n_j,k_j}$ for some n_j, k_j.
- and an oriented graph $\mathcal{G}$ with vertex set $\{N_1, \ldots, N_p\}$, describing how the tiny pieces $\{M_1, \ldots, M_p\}$ are glued together to form M, in terms of adjacencies between sides of the M_j.

More precisely, a vertex N_j of $\mathcal{G}$ contains the following information:
– n_j, k_j, and the index of M_j in $\mathcal{D}_{n_j,k_j}$,
– indexes to the neighbors of N_j in $\mathcal{G}$ (for each side of M_j, the index of the corresponding neighbor and side).
In particular the number of edges in the graph $\mathcal{G}$ is bounded by the total number of sides in the M_j.

We first need an hypothesis ensuring that the additive part of the entropy matches the entropy αn of the class to which M belongs.

Hypothesis 1 *The decomposition is additive in the size parameter (elementary cells are not shared):*

$$n_1 + \ldots + n_p = n + O\left(\frac{n}{\lg n}\right),$$

Observe that we do not require the class $\mathcal{D}_m = \bigcup_k \mathcal{D}_{m,k}$ to have the same entropy as $\mathcal{C}_m$: in particular in our two main examples below we will have $||D_m|| \sim \alpha' m$ with $\alpha' > \alpha$. As a consequence, in order for the representation to be compact we need a second hypothesis on the number of sides.

Hypothesis 2 *The decomposition involves a sub-linear number of sides:*

$$k_1 + \ldots + k_p = O\left(\frac{n}{\lg n}\right).$$

This second hypothesis implies that the whole cost of storing indexes to all the M_i remains of order αn.

Next hypothesis ensures that the elements of $\mathcal{D}$, needed in the decomposition, fit in a small catalog.

Hypothesis 3 *In the decomposition each M_j can be taken of size between $\frac{c}{3} \lg n$ and $c \lg n$, where $c < 1/\alpha'$, with α' the entropy per size unit of $\mathcal{D}$.*

Indeed assume that the indexes are pointing into a table A, containing the explicit representations of all elements of $\mathcal{D}_m$ for $m \leq c \lg n$ for some constant c. If the constant c is chosen small enough, the number of entries in the table is sub-linear: indeed $||\mathcal{D}_m|| = \alpha' m \leq c\alpha' \lg n$, so with $c < 1/\alpha'$, the number of entries in the table is $O(n^{c\alpha'})$. The total storage cost of Table A then remains sub-linear as long as the information for each piece is polynomial in its size $m = O(\log n)$. In particular, explicit answers to local queries can be stored (as local adjacency queries on elementary cells).

2.2 Compactness

Hypothesis 2 above ensures that the graph $\mathcal{G}$ has $O(n/\lg n)$ edges, and Hypothesis 3 that it has $O(n/\lg n)$ vertices. This is already enough to obtain compactness.

Lemma 2 *Under Hypotheses 1, 2, and 3*
the storage of the graph $\mathcal{G}$ requires $O(n)$ bits.

Proof: Recall that each vertex N_j of $\mathcal{G}$ contains: n_j, k_j and the index of M_j in $\mathcal{D}_{n_j,k_j}$, as well as indexes to the neighbors of N_j in $\mathcal{G}$.

As n_j and k_j are smaller than $c \lg n$ and $Kc\lg n$ respectively, we can store them in $2 \lg\lg n + O(1)$ bits.

When summing over the $O(n/\lg n)$ vertices of $\mathcal{G}$ we get a $O\left(\frac{n \lg\lg n}{\lg n}\right)$ bits cost.

Using Equation (1) the index of M_j requires $\alpha n_j + \beta\, k_j \lg n_j + O(\lg n_j)$ bits.

Summing over all M_j yields a global cost of $\alpha \sum_j n_j + \beta(\sum_j k_j) \lg(\max_j n_j) + O\left(\sum_j \lg n_j\right) = \alpha n + O(n/\lg n) + O(n/\lg n)\lg\lg n$.

Using Hypotheses 1 and 2, the cost of all indices to Table A thus reduces to $\alpha n + O\left(\frac{n \lg\lg n}{\lg n}\right)$ bits.

Since there are $O(n/\lg n)$ vertices in $\mathcal{G}$, the index of a neighbor uses $\lg n + O(1)$ bits. Vertex N_j has $O(k_j)$ neighbors, thus the total cost for storing indexes to the neighbors is $(\lg n + O(1)) \cdot \sum_j k_j = O(n)$. Summing all these components yields the claimed complexity. ∎

2.3 Succinctness

In the proof of Lemma 2 the linear part of the storage came from two kinds of contributions: the contribution of indexes in the catalog which is dominated by the entropy αn, and the contribution of the neighboring relations in the graph $\mathcal{G}$. In this section, the cost of this second part is reduced to be sub-linear.

The graph $\mathcal{G}$ is partitioned into *small* pieces gathering $O(\lg n)$ tiny pieces. More precisely we assume that we are able to construct a graph $\mathcal{G}'$ obtained by merging several vertices of $\mathcal{G}$ in a vertex of $\mathcal{G}'$ and linking two such vertices if there is an edge in $\mathcal{G}$ between two of their elements. The vertex set of $\mathcal{G}'$ is $\{N'_1, N'_2, \ldots N'_{p'}\}$ and we denote $|N'_i|$ the number of vertices of $\mathcal{G}$ that have been merged to obtain N'_i and $deg'(N'_i)$ the degree of N'_i in $\mathcal{G}'$.

A vertex N_i' of $\mathcal{G}'$ then consists of the following information:
— $|N_i'|$ and $deg'(N_i')$
— the information for all the vertices $N_j \in N_i'$, stored in a single memory zone.
— indexes to neighbors of N_i' in $\mathcal{G}'$.
The graph $\mathcal{G}'$ has moreover to satisfy the following hypotheses.

Hypothesis 4 *The number of tiny pieces gathered in each small piece N_i' satisfies: $\frac{1}{3}\lg n \le |N_i'| \le \lg n$.*

The number of edges from a given vertex of $\mathcal{G}'$ can be bounded by summing over its elements:

$$deg'(N_i') \le \sum_{N_j \in N_i'} deg(N_j) =$$

$$= \sum_{N_j \in N_i'} k_j \le |N_i'| K c \lg n = O(\lg^2 n)$$

(where K is a constant describing the size of the boundary of tiny pieces, as introduced at Equation (1)). But we need a stronger hypothesis on the total number of edges of $\mathcal{G}'$.

Hypothesis 5 *The number of edges of $\mathcal{G}'$ is linear in its number of vertices:*

$$deg'(N_1') + \ldots + deg'(N_{p'}') = O\left(\frac{n}{\lg^2 n}\right).$$

Lemma 3 *Under Hypotheses 1, 2, 3, 4 and 5 the graph $\mathcal{G}$ can be stored using $\alpha n + O\left(\frac{n \lg\lg n}{\lg n}\right)$ bits.*

Proof: First we notice that in the above representation of $\mathcal{G}$ (Lemma 2), a vertex N_j uses $O(\lg^2 n)$ bits ($O(\lg n)$ bits for the index of M_j in the relevant catalog and $O(\lg n)$ for each of its, at most, $Kc\lg n$ neighbors) which gives easily a bound of $O(\lg^3 n)$ for the memory needed by all vertices of $\mathcal{G}$ that merge in some N_i'.

Hence a local reference to memory addresses relative to some known N_i' requires $\lg\lg n + O(1)$ bits.

Let us now return to the information stored in a vertex $N_j \in N_i'$ of $\mathcal{G}$. To refer to a neighbor N_l of N_j, instead of using an address in the whole memory devoted to $\mathcal{G}$, we refer first to the vertex $N_k' \ni N_l$ and then give the address of N_l in the memory zone devoted to elements of N_k'.

Referring to N_k' is done indirectly by giving its index in the array of the, at most, $O(\lg^2 n)$ neighbors of N_i'. Thus a reference to a neighbor N_l of N_j costs $O(\lg\lg n)$ bits. The analysis of the size of $\mathcal{G}$ is similar to the argument used in Lemma 2: here the cost of a reference to a neighbor does go from $O(\lg n)$ down to $O(\lg\lg n)$ which yields the claimed complexity.

The additional cost for $\mathcal{G}'$ is sublinear since it has $O(\frac{n}{\lg^2 n})$ vertices by Hypothesis 4 and edges by Hypothesis 5 and each costs $O(\lg n)$ bits. ■

2.4 Local planarity and mesh connectivity

The above framework easily applies to trees: a tree with n vertices can be recursively decomposed into tiny trees of logarithmic size, with sides consisting of edges connecting nodes of different tiny trees.

More interestingly Hypotheses 1, 2, 3, 4 and 5 are naturally satisfied when dealing with mesh connectivity (that is, for maps on surfaces). Following for instance [1], a triangulation M with n faces can be partitioned into tiny regions by decomposing an arbitrary dual spanning tree into tiny trees (that is, a spanning tree of the dual graph, connecting all faces accross edges): upon reforming a (local) planar triangulation from each tiny tree, a decomposition $(M_1, \ldots, M_p)$ is obtained, such that:
- each tiny triangulation contains between $\frac{1}{12} \lg n$ and $\frac{1}{4} \lg n$ triangles (that is, $c = 1/4$);
- each triangle belongs to exactly one tiny triangulations;
- boundary edges are regrouped into *sides* (sequences of edges separating it from the same tiny triangulations).

The bound on the number of edges of the graph $\mathcal{G}$ that describes adjacency relations between sides of tiny triangulations follows from Euler's relation.

This direct application of the framework to triangulations using an arbitrary spanning tree forces us, as explained in [1], to consider the catalog $\mathcal{D}_{m,k}$ of all planar triangulations with m edges and one boundary cycle, which is divided into k sides. The entropy of this class of triangulation with a boundary is however $\alpha = 2.17$ bits per face (plus a $\beta k \lg m$ term to take into account the number of sides into which the boundary is divided). As a consequence, the additive part of the entropy leads to a representation which is compact, but succinct only for class of triangulations with a boundary (recall that the class of triangulations of the sphere has only entropy 1.62 bits per triangle).

In summary, one can expect the general framework to yield easily compact representations for mesh connectivity with bounded face degrees. However, as we shall see, more care is needed to choose the decomposition in order to produce a succinct representation.

Local adjacency queries in $O(1)$ time

It still remains to observe that the multi-level structure (described by graphs $\mathcal{G}$ and $\mathcal{G}'$, together with the information associated with tiny pieces M_j), does allow to perform efficiently some local adjacency queries (neighboring queries on elementary cells).

Lemma 4 *Let us consider an object M with a decomposition $[(M_1, \ldots, M_p); \mathcal{G}; \mathcal{G}']$ as defined above. If each of the tiny pieces M_j is a planar connected map having all faces of constant degree and a boundary of arbitrary linear size $O(n_j)$, then it is possible to answer in $O(1)$ time whether two elements of M (vertices or faces) are adjacent or not.*

Proof: The idea is that elementary cells (faces, vertices, edges) are shared by at most one or two adjacent tiny pieces (because of the definition of graph $\mathcal{G}$), except for a small set of *multiple cells* (shared by an arbitrary number of tiny pieces). The planarity of graph $\mathcal{G}$ ensures that the number of multiple cells is $O(\frac{n}{\lg n})$, hence adjacency queries involving multiple cells can be answered with an additional information, which requires in overall a negligible amount of extra storage.

Intuitively, local queries involving only a tiny piece M_i are answered looking at the information stored in Table A.

Adjacency queries concerning elementary cells incident to the boundary of a tiny piece M_j, rely instead on the information in graphs $\mathcal{G}$ and possibly $\mathcal{G}'$. Some care must be taken to deal with the fact that a given cell at the meeting point between a lot of tiny pieces can have a non constant number of representations: however the number of such special cells is negligible and they can be detected at the $\mathcal{G}'$ level. ∎

3 Representing 3-connected planar graphs

In this section section we describe a particular catalog of tiny quadrangulations and an algorithm to decompose any irreducible quadrangulations into tiny regions taken from this catalog. This catalog satisfies Equation 1 so that the framework yields a succinct representation.

Since irreducible quadrangulations are just another representation of 3-connected planar graphs, the result holds for these maps as well.

Preliminaries on 3-connected graphs, quadrangulations and trees

Plane trees are planar maps with only one face, the outer one. In other terms, plane trees only differ from the classical ordered trees in the fact that they are not rooted. A binary tree is a plane tree such that each vertex has 3 neighbors (and hence 2 sons if rooted). A vertex is a leaf if it has degree 1, otherwise it is an internal node.

Edges incident to leaves are called *stems*, remaining edges are called *inner edges*. A quadrangulation is a planar map having all its faces of degree 4. A dissection of the hexagon by quadrangular faces is a planar map whose outer face has degree 6 and inner faces have degree 4. A quadrangulation or dissection of the hexagon by quadrangular faces is said irreducible if it has no separating 4-cycle (we also call them *irreducible dissections*).

The following construction is more or less a folklore variation on the standard duality construction for planar maps: given a 3-connected planar map M, color its vertices in black and put a white vertex in the middle of each face and triangulate each face from this white vertex. The resulting new edges form a quadrangulation Q (each face is made of the two triangles incident to an edge of M). It is not difficult to check that the 3-connectedness of M is equivalent to the irreducibility of Q. Finally let us observe that it is possible to associate a rooted dissection d of the hexagon to a rooted irreducible quadrangulation Q by deleting the root edge.

Intuitively, a quadrangulation Q can be viewed as an implicit representation of a planar map, induced with a bicoloration of its vertices: white (resp. black) vertices of Q stand for faces (resp. vertices) of the original (resp. dual) map (see Fig. 1).

More precisely, testing adjacency relations between vertices and faces in map M corresponds to answer neighboring queries on vertices incident to the same face in the

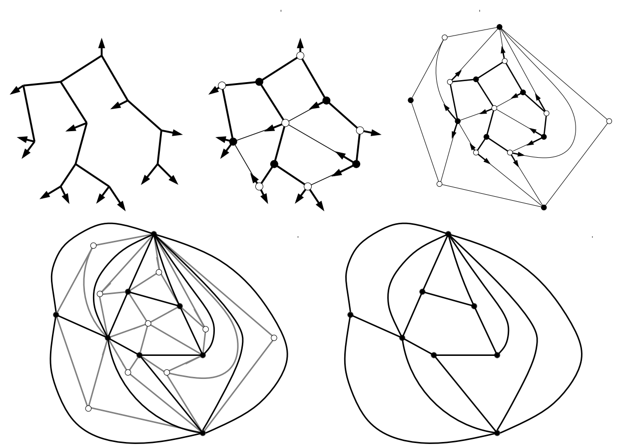

Figure 1: First Pictures describe the closure defining the correspondence between plane rooted binary trees and irreducible dissections of the hexagon. The dissection and the corresponding 3-connected graph are also shown. Black circles (resp. white circles) represent vertices of the primal (dual) graph.

quadrangulation Q. It will prove convenient to describe our construction in term of quadrangulations.

3.1 Decomposition of the quadrangulation

Our decomposition strategy will take advantage of the fact that irreducible dissections admit a special class of canonical spanning trees, introduced in [8]:

Proposition 5 *There exists a bijection between the class of binary trees on n internal nodes and the class of irreducible dissections with n internal vertices, which can be computed in $O(n)$ time.*

Let us suppose to have an irreducible dissection Q of the hexagon with quadrangular faces. Following the traversal strategy explained in [8], it is possible to perform an opening algorithm on Q which returns a binary tree: the result is a vertex spanning tree of Q, whose complete closure [8, Lemma 2] is exactly the original dissection of the hexagon (corresponding to a 3-connected planar graph). More precisely it is a binary tree B on n nodes having $n+2$ stems and $(n-1)$ inner edges (the correspondence is illustrated in Figure 1).

Let us recall a previous result concerning tree decompositions, stated in the following Lemma[16]:

Lemma 6 *Given a binary tree $\mathcal{B}$ on n nodes and a positive integer parameter δ, we can produce in linear time a partition of $\mathcal{B}$ into a family of sub-trees $\mathcal{B}_j$, whose sizes satisfy $\delta \leq \|\mathcal{B}_j\| \leq 3\delta$.*

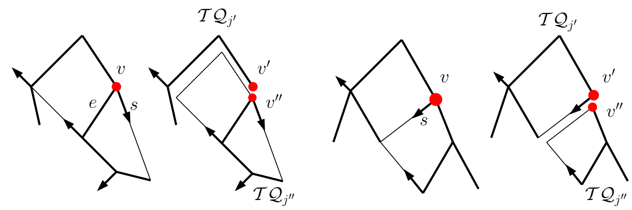

Figure 2: These Pictures explain how distributing stems and (dummy) faces between tiny quadrangulations sharing a node v (a multiple node, represented with a small circle).

Applying several times the algorithm above, we obtain a partition of $\mathcal{B}$ into small binary trees having between $\frac{1}{3}\lg^2 n$ and $\lg^2 n$ nodes, which are then decomposed into tiny binary trees having between $\frac{1}{30}\lg n$ and $\frac{1}{10}\lg n$ nodes. Such a decomposition form a partition of the edges of $\mathcal{B}$ (which are edges of the original quadrangulation). Let us observe that only nodes, those which are roots of tiny (resp. small) trees, are shared by different tiny (resp. small) trees: each of these nodes (having degree d) is split into a degree $d-1$ root of one tree (*descendent* tree) and a leaf of another tree (*ancestor* tree). The way we have partitioned $\mathcal{B}$ guarantees that number of tiny (resp. small) trees is $\Theta(\frac{n}{\lg n})$ (resp. $\Theta(\frac{n}{\lg^2 n})$).

3.2 Correspondence between tiny quadrangulations and tiny trees

The aim of this section is to describe a canonical way of obtaining a decomposition $\{M_1, \ldots, M_p\}$ of $\mathcal{Q}$, starting from the decomposition of the vertex spanning tree $\mathcal{B}$. Here we suppose we are given one tiny tree, denoted $\mathcal{TB}_j$, having $n_j \leq \frac{1}{10}\lg n$ nodes and w_j false stems, obtained by decomposing $\mathcal{B}$: we are going to show how to produce a tiny quadrangular map of size $\Theta(\lg n)$ from $\mathcal{TB}_j$. A tiny tree $\mathcal{TB}_j$ has n_j nodes and $n_j + 2$ stems: we now distinguish two kinds of leaves. Firstly there may exist some leaves which were already leaves in the original tree $\mathcal{B}$. After the decomposition algorithm is performed, there are now some leaves which was inner nodes in $\mathcal{B}$ before we split them: their incident edges are now called *false stems*. Recall that tiny trees are rooted (roots are shared by different tiny trees), and the number of false stems in a tiny tree is not fixed and it ranges in $0 \ldots n_j + 2$, depending on the way $\mathcal{B}$ has been partitioned.

3.3 Local closure

Now it suffices to apply a "local closure" algorithm, inspired to the one introduced in [8], whose main steps are listed below (recall that in the original algorithm of [8], there was no distinction between true and false stems).

Let us perform a ccw traversal of $\mathcal{TB}_j$, starting from its root and walking along its edges (inner edges, stems and false edges).

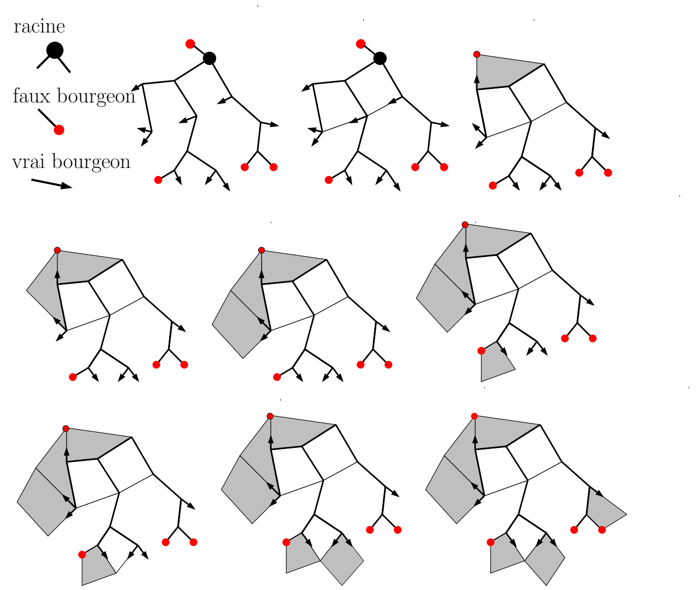

Figure 3: Our local closure. The result of the closure operation does depend on the distribution of false stems. In our example, the binary tree has 11 inner vertices, $11 + 2$ leaves (true and false stems, including the stem incident to the root), and can be optimally encoded by the binary word of length $2 \cdot 11$: 1101001011001001011000; while the corresponding false stems distribution is defined by the bit-vector of length 13 and weight 4: 0000100001101.

– When traversing a true stem, which is preceded by 3 internal nodes (and not stems), its local closure consists in linking its incident node, with the preceding third node (on the boundary of the outer face) to create a quadrangular face (a white face in Figure 3).

– We do not perform the merging of false stems (drawn with small circles in Figure 3).

– When traversing a (true) stem s, not preceded by 3 true nodes (original nodes existing in $\mathcal{TB}_j$), its local closure consists in attaching a *dummy quadrangular face* (a grey face in Fig. 3) to the boundary of $\mathcal{TQ}_j$, in such a way that the dummy face is then incident to the stem s and does not enclose a false stem nor a *dummy edge* (i.e. an edge incident to a dummy face) previously added. Similarly, vertices incident to dummy faces, and not existing in the tiny tree, are called *dummy vertices*.

In this way we produce a planar map whose internal faces are all quadrangles, and having an outer face of arbitrary size: inner edges (in the tree) may now be incident twice to the outer face. The planar connected maps obtained in this way are also called *tiny quadrangulations* (the result of the procedure above is shown in Figure 3).

It is easy to observe that all the faces of the initial quadrangulation have been assigned to exactly one tiny quadrangulation, as they are incident each to one true stem.

Once the initial spanning tree $\mathcal{B}$ has been decomposed, we have to specify a rule for assigning (true) stems incident to nodes shared by tiny trees. We proceed as follows, assuming that v is shared by two tiny trees $\mathcal{TB}_{j'}$ and $\mathcal{TB}_{j''}$, and denoting the possibly incident stem by s (see last Pictures in Fig. 1):

- if there exists in the descendant tree $\mathcal{B}$ an inner edge e incident to v to the left of s, then we attach the stem s to the tiny sub-tree $\mathcal{TB}_{j''}$ having v as root and containing e;

- otherwise s is the leftmost sibling of node v in $\mathcal{B}$, and we do attach s to the ancestor tree $\mathcal{TB}_{j'}$.

3.4 How does our framework apply

Lemma 7 *Given a quadrangulation $\mathcal{Q}$ with n vertices, our new splitting strategy produces a decomposition into tiny quadrangulations $\{\mathcal{TQ}_1, \dots, \mathcal{TQ}_p\}$ satisfying Hypotheses 1, 2, 3, 4 and 5, and hence yields a succinct representation of $\mathcal{Q}$ requiring asymptotically $2n + o(n)$ bits.*

Proof: The additivity hypothesis holds, since (true) nodes in tiny quadrangulations (nodes of the tiny spanning tree) are not shared by tiny quadrangulations.

The quadrangulation $\mathcal{Q}$ is decomposed into tiny quadrangulations $\mathcal{TQ}_j$ each containing between $\frac{1}{30}\lg n$ and $\frac{1}{10}\lg n$ nodes (nodes of the corresponding tiny vertex spanning tree): here the constant c introduced in Section 2 is set to $\frac{1}{10}$.

The graph $\mathcal{G}$ used to describe adjacency relations between tiny quadrangulations is a planar map (each tiny quadrangulation is a connected planar map, whose edges may be incident twice to the outer face, and then doubly counted as boundary edges) having faces of degree at least 3 (for example, a degree 2 face incident to multiple edges can be contracted, observing that the corresponding sides are consecutive and shared by the same two tiny quadrangulations).

Hence Euler's relation ensures that the number of arcs of $\mathcal{G}$ (and hence the number of sides) is $O(\frac{n}{\lg n})$. Hypotheses 1, 2, 3 5 and 4 are satisfied by $\{\mathcal{TQ}_1, \dots, \mathcal{TQ}_p\}$, hence Lemma 3 yields a succinct representation for the class of quadrangulations achieving the optimal asymptotic bound of 2 bits per vertex.

Here an object $M_j = \mathcal{TQ}_j$ is a *tiny quadrangulation*: $\mathcal{TQ}_j$ is obtained via our local closure from a tiny tree $\mathcal{TB}_j$ having n_j nodes, $n_j + 2$ stems and w_j false stems; moreover $\mathcal{TQ}_j$ is induced with a partition of its boundary edges into k_j sides[1].

A tiny colored quadrangulation is completely specified by: a Dyck word of length $2n_j$ (for the binary spanning tree), a binary word of length $n_j + 2$ and weight w_j (describing false stems) and a binary word of length $4n_j + 6$ and weight k_j (describing the partition of boundary edges of $\mathcal{TQ}_j$ into sides).

[1]One can easily see that given a tiny binary tree with m nodes, the corresponding tiny quadrangulation has a boundary of size at most $4m+6$. Concerning inner edges, doubly incident to the outer face, their number is $m-1$, hence their contribution to the boundary size is at most $2(m-1)$. On the other hand, the contribution of dummy faces is maximum when a dummy face is produced by a true stem immediately preceded by a false stem: as there are $m+2$ stems, there may be at most $\lceil \frac{m+2}{2} \rceil$ dummy faces, each contributing for 4 edges. Finally the size of the boundary of a tiny quadrangulation is bounded by $2(m-1)+4(\frac{m+2}{2}+1) = 4m+6$.

Hence $\mathcal{TQ}_j$ can be encoded by a reference to an element in $\mathcal{D}$, whose cost is $2n_j + w_j \lg(n_j + 2) + k_j \lg(4n_j + 6) \leq 2n_j + \beta k_j(\lg n_j + O(1))$ (as $w_j \leq k_j$).

The constant c can be chosen so that the Table A containing the explicit representations of the elements of $\mathcal{D}$, requires $o(n)$ bits (recall that $n_j \leq \frac{1}{10} \lg n$). ∎

3.5 Representing 3-connected planar graphs

Given a 3-connected graph with e edges, we first design a succinct representation of the associated quadrangulation. Since the corresponding vertex spanning tree has $e-5$ inner nodes (vertices of the primal and dual graph) our representation requires asymptotically $2(e-5)+o(e) = 2e+o(e)$ bits, according to Lemma 7. It suffices to observe that testing adjacency between two vertices (resp. two faces) in a 3-connected graph, is equivalent to check if two black (resp. white) nodes are opposite in the same quadrangular face, in the associated quadrangular map $\mathcal{Q}$. This operation can be efficiently performed in $O(1)$ time with a slightly modification of the standard adjacency queries considered in Lemma 4 (see Section A for more details).

3.6 Unique representations for vertices

One major problem to solve concerns the representation of vertices, which is not unique. As already observed, a number of elementary cells (*multiple cells*) are shared by several tiny pieces. One possible solution consists in exploiting the correspondence between vertices in the quadrangulation and nodes in the spanning tree. It simply suffices to associate to each vertex in $\mathcal{Q}$ the corresponding node in the tree $\mathcal{B}$.

Then inner nodes in a tiny tree $\mathcal{TB}_j$ are uniquely specified. Remaining vertices, multiple vertices shared by tiny pieces, can be uniquely identified, saying that their *canonical representative* is the corresponding node in an adjacent tiny piece (in particular, for a multiple node shared by two tiny trees, its canonical representative is the leaf node in the ancestor tree).

4 A succinct representation for planar triangulations

Preliminaries on planar triangulations and trees

As done for 3-connected graphs, we take advantage of a recent bijection between planar triangulations and trees introduced in [17].

Proposition 8 *There exists a $2n$-to-2 correspondence between the class of plane trees with n nodes having 2 leaves per node, and the class of rooted planar triangulations with $n+2$ vertices.*

This correspondence relies on an *opening/closure algorithm* which computes a special vertex spanning tree (with two leaves per node) on n nodes from a triangulation $\mathcal{T}$, induced with its minimal realizer.

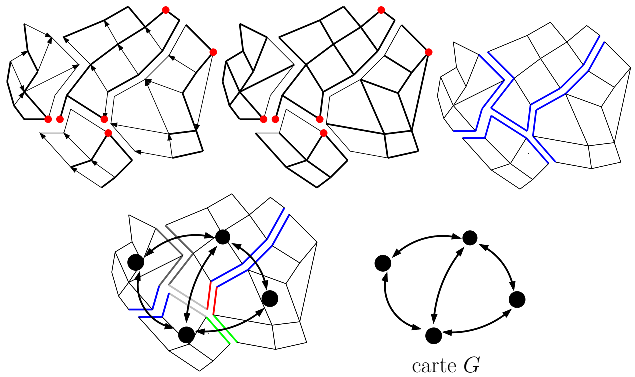

Figure 4: This Figure illustrates our multi-level hierarchical representation of a quadrangulation. The decomposition of $\mathcal{B}$ provides, via our local closure, a partition of $\mathcal{Q}$, into tiny quadrangulations. Neighboring relations between tiny quadrangulations are described by a planar map G.

4.1 Decomposition of the graph

Here we suppose we are given a rooted triangulation $\mathcal{T}$ with $n+2$ vertices and the corresponding vertex spanning tree $\mathcal{B}$ on n nodes, whose *complete closure* is the initial triangulation (according to the closure/opening algorithm introduced in [17]). Since there are no restrictions on the tree $\mathcal{B}$ (it is just an ordered tree, with two leaves per node), we cannot in general decompose $\mathcal{B}$ to get a partition into sub-trees (as done for binary trees). We then make use of a useful strategy for decomposing ordered trees [9], expressed by the following Lemma:

Lemma 9 *Given a $\mathcal{B}$ on n nodes and $\delta \geq 2$, we can compute a family of sub-trees that is a covering of $\mathcal{B}$. Their sizes satisfy $\mathcal{B}_j \leq 3\delta - 4$ (and $\mathcal{B}_j \geq \delta$, if the $\mathcal{B}_j$ do not contain the root of $\mathcal{B}$).*

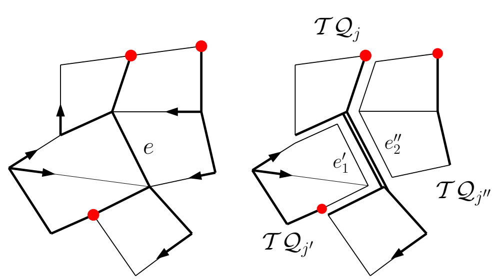

Figure 5: These two Pictures illustrate the adjacency relations between tiny quadrangulations: since edges are allowed to be incident twice to the outer face, two tiny quadrangulations $\mathcal{TQ}_{j'}$ and $\mathcal{TQ}_{j''}$ may be not incident even if ”sharing” an edge e.

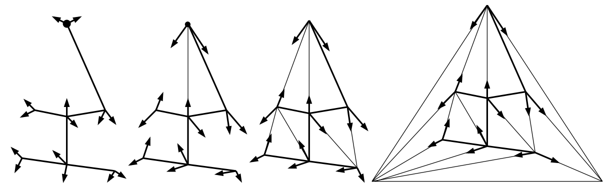

Figure 6: Vertex spanning tree bijection between rooted planar triangulations and rooted trees with 2 leaves per node.

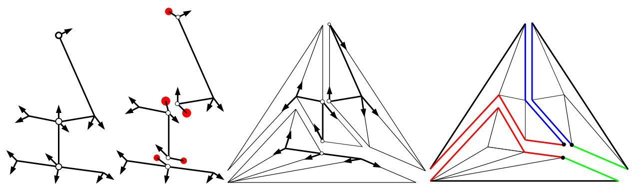

Figure 7: In these Pictures is depicted our strategy for constructing a decomposition of the original triangulation $\mathcal{T}$ into tiny triangulations. At first the vertex spanning tree of $\mathcal{T}$ is decomposed into tiny trees. We perform our local closure algorithm on tiny trees, producing tiny triangulations. Each tiny triangulation is provided with a partition into sides of its boundary edges describing neighboring relations.

Applying several times Lemma 9 to $\mathcal{B}$, we obtain a family of small sub-trees covering $\mathcal{B}$ of size $\Theta(\lg^2 n)$, which are then decomposed into tiny sub-trees having $\Theta(\lg n)$ nodes. This family of sub-trees forms a cover of the nodes of $\mathcal{B}$ such that two sub-trees can intersect only at their root.

4.2 Correspondence between tiny trees and tiny triangulations

As done for quadrangulations, it is possible to define a local *closure algorithm* (inspired to the one described in [17]), providing a correspondence between tiny trees and tiny triangulations. Firstly, if a tiny tree $\mathcal{TB}_j$ has an inner node v of degree 3, which appears as root of one (or more) different tiny tree $\mathcal{TB}_{j'}$ (neighbor of $\mathcal{TB}_j$), we set v as *false degree* 3 *inner node* (since v had degree more than 3 in $\mathcal{B}$). Observe that tiny trees could have less than two leaves per node. In order to make a tiny tree belong to the same class as $\mathcal{B}$, we perform some modifications on it. Original stems in $\mathcal{B}$ are duplicated and distributed between tiny trees $\mathcal{TB}_j$ so that each node has two leaves: duplicated stems are called false stems and are assigned to the tiny trees sharing a same node v (the cyclic order of neighbors around v must be respected). Observe that false inner nodes are incident to at least one false stem, at the exception of the root node of a tiny tree (see Figure 7). It is straightforward to observe that the number of false degree 3 inner nodes and false (duplicated) stems is in overall $O(\frac{n}{\lg n})$. Our local closure of a tiny tree $\mathcal{TB}_j$ start by performing a ccw traversal along its edges. As done in Section

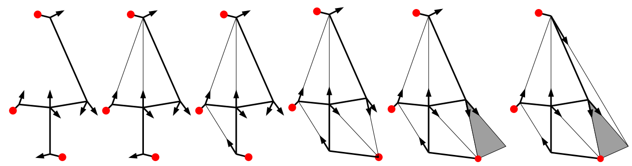

Figure 8: This Figure shows a tiny triangulation obtained via our local closure from a tiny tree $\mathcal{TB}$. In our example we start with a tiny tree having 5 nodes and 10 leaves: there are 3 false stems (small circles) and 3 false inner nodes (including the root). The stems distribution is described by a binary word of length 10 and weight 3: 0010100001.

3, we add a (true or dummy) triangular face by performing the merging of true stems, depending on the distribution of false stems and false inner nodes. More precisely, if a true stem s is preceded by 2 (true) inner nodes, we link its incident node to the second preceding node to create a triangular face. Otherwise, if s is preceded (on the boundary of the outer face) by a vertex v which is not an inner node we add a *dummy* triangular face incident to s and v (a grey face in Figure 8). The planar connected maps $\mathcal{TT}_j$ we obtain in this way (whose internal faces are all triangles, and with an outer face of arbitrary size) are called *tiny triangulations*.

Lemma 10 *Given a planar triangulation $\mathcal{T}$ with $n+2$ vertices, our new splitting strategy produces a decomposition $\{\mathcal{TT}_1, \ldots, \mathcal{TT}_p\}$ into tiny colored triangulations satisfying the Hypotheses 1, 2, 3, 4 and 5, and hence yields a succinct representation of $\mathcal{T}$ requiring asymptotically $3.24n + o(n)$ bits.*

Proof: Our arguments rely on the same remarks used in the proof of Lemma 7. The additivity Hypothesis holds, since tiny triangulations only share duplicated nodes (roots of tiny trees), whose number is $O(\frac{n}{\lg n})$. Here an object $M_j = \mathcal{TT}_j$ is a tiny triangulation: $\mathcal{TT}_j$ is obtained via our local closure from a tiny tree $\mathcal{TB}_j$ having n_j nodes, $2n_j$ stems and w_j false stems; its boundary edges are partitioned into k_j sides. Hence a tiny triangulation is described by: a binary word of length $4n_j - 2$ and weight $n_j - 1$ (there are about $2^{3.24n_j}$ such words, see [17]), a binary word of length $2n_j$ and weight w_j (for false stems) and a binary word of length $6n_j - 2$ and weight k_j (a tiny triangulation has at most $6n_j - 2$ boundary edges). Again the constant c can be chosen so that Catalog $\mathcal{D}$ requires an asymptotic negligible amount of space. ■

5 Concluding remarks

We have presented a general framework for describing succinct representations of planar maps. In the particular case of 3-connected graphs and triangulations, we propose moreover canonical decompositions which, combined with the general framework, yield encodings that achieve asymptotically the information theory optimal bound for

Algorithm	triangulated	3-connec.
Munro Raman (Focs 97)	$2e + 8n$ or $7m$	$2e + 8n$
Chuang et al. (Icalp 98)	$2e + n$ or $3.5m$	$2e + 2n$
Chiang et al. (Soda 01)	$2e + 2n$ or $4m$	$2e + 2n$
Castelli Aleardi et al.	$2.175m$	-
our new encodings	$1.62m$	$2e$

Table 1: Comparison of existing compact representations for simple planar graphs, with e edges, m faces and n vertices (lower order terms are omitted).

the storage, while supporting efficiently standard local navigation operations. The generality of our arguments suggests that our framework could apply to other popular encoding schemes, getting compact representations of other classes of planar graphs. This should take advantage of the similarities between explicit spanning tree coding [12, 20] and region-growing approaches [18, 19] as discussed in [10]. It is still open the problem of extending optimal encodings [17] and compact representations of graphs [7, 6] to the case of higher genus triangulated surfaces. As done in previous works on compact representations [6, 7, 15], our results here are manly theoretical: although we do not intend to implement the our framework in the presentation given above, this work could be a good source of inspiration for practical structures. In practice, n could be considered more as a constant than a parameter: we intend to design a space efficient structure whose aim is to push the swapping limit of triangulation, say from 10 Mvertices to 40 Mvertices. Such a structure can be based on a splitting in tiny and small triangulations whose sizes are limited by finely tuned constants.

Acknowledgments We are grateful to Eric Fusy for enlightening discussions. First author would like to thank the Embassy of France in Rome and the University of Milan for partially supporting this work.

References

[1] L. Castelli Aleardi, O. Devillers, and G. Schaeffer. Succinct representation of triangulations with a boundary. In *Proc. of WADS*, 134-145, 2005.

[2] L. Castelli Aleardi, O. Devillers, and G. Schaeffer. Dynamic update of succinct triangulations. In *Proc. of CCCG*, pages 135-138, 2005.

[3] P. Alliez and C. Gotsman. Recent advances in compression of 3d meshes. In N.A. Dodgson, M.S. Floater, and M.A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 3–26. Springer-Verlag, 2005.

[4] D. Blanford, G. Blelloch, and I. Kash. Compact representations of separable graphs. In *Proc. SODA*, 342-351, 2003.

[5] J.-D. Boissonnat, O. Devillers, S. Pion, M. Teillaud, and M. Yvinec. Triangulations in CGAL. *Comp. Geom.: Theory and Appl.*, 22:5–19, 2002.

[6] Y.-T. Chiang, C.-C. Lin, and H.-I. Lu. Orderly spanning trees with applications to graph encoding and graph drawing. *Proc. of SODA*, pages 506–515, 2001.

[7] R.C.-N Chuang, A. Garg, X. He, M.-Y. Kao, and H.-I. Lu. Compact encodings of planar graphs via canonical orderings and multiple parentheses. *Proc. of ICALP* , pages 118–129, 1998.

[8] E. Fusy, D. Poulalhon and G. Schaeffer. Dissections and trees, with applications to optimal mesh encoding and to random sampling In *Proc. of SODA*, 690-699, 2005.

[9] R. Geary, R. Raman and V. Raman. Succint ordinal trees with level-ancestor queries. In *Proc. of SODA*, 1-10, 2004.

[10] M. Isenburg and J. Snoeyink. Graph Coding and Connectivity Compression. manuscript, 2004.

[11] G. Jacobson. Space efficients static trees and graphs. In *Proc. of FOCS*, 549–554, 1989.

[12] K. Keeler and J.Westbrook. Short encodings of planar graphs and maps. *Discr. Appl. Math.*, 239-252, 1995.

[13] A. Khodakovsky and P. Alliez and M. Desbrun and P. Schroder. Near-Optimal Connectivity Encoding of 2-Manifold Polygon Meshes. In *J. of the Graph. Models*, pages , 2002.

[14] M. Kallmann and D. Thalmann. Star-vertices: a compact representation for planar meshes with adjacency information. *J. of Graphics Tools*, 6:7–18, 2002.

[15] J. I. Munro and V. Raman. Succint representation of balanced parantheses and static trees. *SIAM J. on Computing*,31:762-776,2001. (also in FOCS '97).

[16] J. I. Munro, V. Raman, and A. J. Storm. Representing dynamic binary trees succintly. *Proc. of SODA*,529-536,2001.

[17] D. Poulalhon and G. Schaeffer. Optimal coding and sampling of triangulations. In *Proc. of ICALP*, 1080–1094,2003.

[18] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. In *IEEE Trans. Visual. and Comp. Graph.*, 5:47-61, 1999.

[19] C. Touma and C. Gotsman Triangle mesh compression. In *Graphics Interface*, pages 26-34, 1998.

[20] G. Turan. Succinct representations of graphs. In *Discr. and Appl. Math*, pages 289-294, 1984.

[21] W. Tutte. A census of planar triangulations. *Canadian J. of Mathematics*, 14:21-38, 1962.

[22] W. Tutte. A census of planar maps. *Canadian J. of Mathematics*, 15:249-271, 1963.

A Local queries in the quadrangulation

A vertex in a tiny quadrangulation $\mathcal{TQ}_j$ can be specified by a triple (N'_i, a, v): where N'_i is a node of map $\mathcal{G}'$, corresponding to the small piece $\mathcal{SQ}_i$ to which it belongs, a is a local pointer to the zone of memory related to node the N_j, and v is the index of the vertex in the explicit representation, stored in Table A, corresponding to $\mathcal{TQ}_j$. As already observed in [1], in our encoding vertices may be not uniquely represented, because they are shared by different tiny pieces. Analogously, representing (quadrangular or triangular) faces is done specifying a triple (N'_i, a, f): their representation is uniquely defined, as faces are not shared by tiny pieces.

Adjacency queries between faces and vertices

Next Lemma provides an useful tool for local navigation between faces of adjacent tiny pieces: its proof relies on arguments similar to the one introduced in [1], which are still valid for arbitrary tiny pieces (having faces of arbitrary constant degree).

Lemma 11 *Given an object $\mathcal{M}$ and its decomposition into tiny (and small) pieces $[(M_1, \ldots, M_p); \mathcal{G}; \mathcal{G}']$, it is possible to answer in constant time the following local queries:*

- $Neighbor((N'_i, a, f), (N'_i, a, e))$: given a face f and an edge e incident to f, it returns the face $(N'_{i'}, a', f')$ adjacent to f containing e.

- $Adjacent((N'_i, a, v), (N'_{i'}, a', w))$: says if two vertices v and w are adjacent.

Unique representation for vertices

It is possible to avoid the problem of having multiple representations of vertices, by adopting a local labelling scheme and distinguishing vertices into 3 categories: vertices internal to a tiny piece, vertices shared by more than 2 tiny pieces, and vertices shared by more than 2 small pieces. As observed in Section 3.2, we can associate to each vertex a unique representation in a canonical way, using the correspondence between vertices in the original graph (quadrangulation or triangulation) and nodes in the vertex spanning tree. Next Lemma allows to deal with multiple vertices:

Lemma 12 *It is possible to answer in $O(1)$ time the following queries involving multiple vertices, using asymptotically $o(n)$ extra bits:*

– $Same((N'_i, a, v), (N'_{i'}, a', w))$: says if v and w represent the same vertex in the graph;

– $Node(N'_i, a, v)$: returns the canonical representative of vertex v (a triple indicating the tiny and small sub-pieces to which v belong as node).

Local adjacency queries on the quadrangulation

Concerning the local navigation in the quadrangulation, our representation allows to perform in $O(1)$ time the following operation (which provides efficient navigation in the original 3-connected graph, as discussed in Section 3.5).

Lemma 13 *In a quadrangulation $\mathcal{Q}$ it is possible to answer in $O(1)$ time the following neighboring query:*

- $Opposite(v, w)$: returns true if two vertices v and w are opposite and incident to the same quadrangular face in $\mathcal{Q}$.

Proof: The validity of our arguments relies on the following properties that hold for 3-connected planar graphs and corresponding quadrangular irreducible dissections.

- The vertex spanning tree introduced in [8] is a binary tree, implying that the degree of nodes and the number of stems per node is bounded and constant.

- Every quadrangular face in the decomposition of $\mathcal{Q}$ belongs exactly to one tiny quadrangulation. Moreover every node v is incident to at most two dummy faces in the same tiny quadrangulation: for nodes with two stems, we observe that each dummy face is created by merging one of its stems; for nodes with one stem (producing one dummy face), only one more dummy face may exist, possibly incident to the descendant node in the tree (see node w in Figure 9).

- For each pair of vertices v and w there is at most one quadrangular face (if it exists) containing the two vertices and for which v and w are opposite (because the quadrangulation $\mathcal{Q}$ has no separating 4-cycle).

Let be q the quadrangular face possibly incident v and w, let us call $v', v'', \ldots$ (resp. $w', w'', \ldots$) their copies, and let $\mathcal{TQ}_j$ and $\mathcal{TQ}_{j'}$ denote the tiny quadrangulations containing them. We may suppose, without loss of generality, that v precedes w on the boundary of the spanning tree $\mathcal{B}$, which is ccw oriented. For the sake of clarity, let us assume that v and w are not multiple nodes, roots of different tiny trees (as in Figure 9). If v and w are vertices lying in the same tiny quadrangulation (hence $j = j'$) we can answer by simply looking at the explicit representation stored in Table A: as v and w are at distance 2 in $\mathcal{TQ}_j$, answering this query requires $o(n)$ extra bits. If the vertices belong to different tiny quadrangulations (not necessarily adjacent tiny quadrangulations) we retrieve at first their canonical representatives, v and w using the function *Node*.If v and w belong to the same tiny quadrangulation, we can repeat the test used in the case above. Otherwise we proceed as follows, denoting by q the quadrangular face possibly existing between v and w. If v and w are opposite and incident to a face q, then q must belong to the same tiny quadrangulation containing w and be incident to one of the edges of $\mathcal{TB}_{j'}$ (inner edge or stem) which is incident to w. Since the degree of nodes in $\mathcal{B}$ is bounded we know that there exist at most 2 quadrangular faces satisfying the last condition, say q_1 and q_2 (grey faces in Figure 9). Let us now retrieve the two vertices v'' and x'', opposite to w, which are incident to q_1 and q_2. Finally it suffice to test whether the canonical representative of v'' and x'' do coincide with v (canonical representative of the copies of v). For concluding, we observe that the case where v (or w) is a multiple node can be dealt in a similar manner: we can repeat the steps above a constant number of times, since each multiple node is contained in at most two different tiny trees. ■

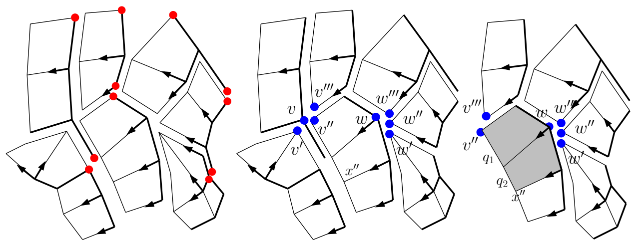

Figure 9: In these Pictures is depicted the case of two vertices v and w, opposite and incident to the same quadrangular face: v and w are multiple vertices (shared by different tiny pieces), but not multiple nodes (as they do not belong to different tiny trees). Multiple nodes are represented in the first Picture with small circles.