Algorithms for graphs on surfaces: from graph drawing to graph encoding

Luca Castelli Aleardi

Université Paris Cité

MÉMOIRE D'HABILITATION À DIRIGER DES RECHERCHES

(spécialité Informatique)
Université Paris Cité

ALGORITHMS FOR GRAPHS ON SURFACES: FROM GRAPH DRAWING TO GRAPH ENCODING

LUCA CASTELLI ALEARDI

Soutenue publiquement le 27 juin 2022 en visio-conférence devant le jury composé de:

Nicolas BONICHON	MdC, HDR	Université de Bordeaux	
Erin WOLF CHAMBERS	Professeur	Saint Louis University	(rapporteur)
Giuseppe DI BATTISTA	Professeur	Università Roma Tre	(rapporteur)
Stefan FELSNER	Professeur	Technische Universität Berlin	(rapporteur)
Claire MATHIEU	DR CNRS	Université Paris Cité	

COMPLETE PUBLICATION LIST

Most results presented in this manuscript are not new and correspond to works published after my PhD. The results obtained during my PhD are not included in this manuscript.

PUBLICATIONS OBTAINED AFTER MY PHD

Journals

[J6] L. Castelli Aleardi, É. Fusy and O. Devillers. Canonical ordering for graphs on the cylinder, with applications to periodic straight-line drawings on the flat cylinder and torus. *J. of Computational Geometry (JoCG)*, 9(1): 391-429, 2018. — Chap. 3, 5

[J5] L. Castelli Aleardi and O. Devillers. Array-based compact data structures for triangulations: Practical solutions with theoretical guarantees. *Journal of Computational Geometry (JoCG)*, 9(1): 247-289, 2018. — Chap. 4

[J4] J. Barbay, L. Castelli Aleardi, M. He and J. I. Munro, Succinct Representation of Labeled Graphs. *Algorithmica* 62(1-2): 224-257 (2012). — /

[J3] L. Castelli Aleardi, É. Fusy and T. Lewiner. Schnyder woods for higher genus triangulated surfaces, with applications to encoding. *Discrete and Computational Geometry (special issue of SoCG 2008)*, 32(3): 489-516, 2009. — Chap. 3, 4

Book chapters

[B1] L. Castelli Aleardi, O. Devillers, and J. Rossignac. Triangulation Data Structures. In *Encyclopedia of Algorithms 2016*, Springer, p.2262-2267, 2016. — Chap. 4

International conferences

[C15] L. Castelli Aleardi. Balanced Schnyder woods for planar triangulations: an experimental study with applications to graph drawing and graph separators. In *Proc. of 27th Int. Symposium on Graph Drawing and Network Visualization (GD 2019)*, Springer LNCS, vol. 11904, p. 114-121, 2019. — Chap. 6

[C14] L. Castelli Aleardi, S. Salihoglu, G. Singh and M. Ovsjanikov. Spectral Measures of Distortion for Change Detection in Dynamic Graphs. In *Proc. of 7th Int. Conference on Complex Networks and Their Applications (Complex Networks 2018)*, Springer SCI, vol. 813, p. 54-66, 2018. — /

[C13] L. Castelli Aleardi, G. Denis and É. Fusy. Fast spherical drawing of planar triangulations: an experimental study of graph drawing tools. In *Proc. of 17th Int. Symposium on Experimental Algorithms (SEA 2018)*, LIPIcs, 24:1-24:14, 2018. — Chap. 5

[C12] L. Castelli Aleardi, A. Nolin and M. Ovsjanikov. Efficient and practical tree preconditioning for solving Laplacian systems. In *Proc. of 14th Int. Symp. on Experimental Algorithms (SEA 2015)*, LNCS, vol. 9125, p. 219-231, 2015. — /

[C11] L. Castelli Aleardi, É. Fusy and A. Kostrygin. Periodic planar straight-frame drawings with polynomial resolution. In *Proc. of 11th Latin American Theoretical INformatics Symposium (LATIN 2014)*, LNCS, vol. 8392, p. 168-179, 2014. — Chap. 5

[C10] L. Castelli Aleardi, O. Devillers, and É. Fusy. Canonical ordering for triangulations on the cylinder, with applications to periodic straight-line drawings. In *Proc. 20th Int Symp. on Graph Drawing (GD 2012)*, LNCS, vol. 7704, p. 376-387, 2012. — Chap. 5

[C9] L. Castelli Aleardi, O. Devillers, and J. Rossignac. ESQ: Editable SQuad representation for triangle meshes. In *Proc. of XXV SIBGRAPI - Conf. on Graphics, Patterns and Images (Sibgrapi 2012)*, IEEE Comp. Soc., p. 110-117, 2012. — /

[C8] L. Castelli Aleardi and O. Devillers. Explicit array-based compact data structures for triangulations. In *Proc. of the 22th Int. Symp. on Algorithms and Computation (ISAAC 2011)*, LNCS, vol. 7074, p. 312-323. — Chap. 4

[C7] L. Castelli Aleardi, É. Fusy and T. Lewiner. Optimal encoding of triangular and quadrangular meshes with fixed topology, in *Proc. of the 22nd Canadian Conf. on Comp. Geom. (CCCG 2010)*, p. 95-98, 2010. — Chap. 4

[C6] L. Castelli Aleardi, É. Fusy and T. Lewiner. Schnyder woods for higher genus triangulated surfaces, in *Proc. of the 24th annual ACM Symp. on Computational Geometry (SoCG 2008)*, p. 311-319. — Chap. 3, 4

[C5] J. Barbay, L. Castelli Aleardi, M. He and I. J. Munro. Succinct representations of labeled graphs, in *Proc. of the 18th Int. Sym. on Algorithms and Computation (ISAAC 2007)*, LNCS, vol. 4835, p. 316-328. — /

PUBLICATIONS OBTAINED DURING MY PHD

Journals

[J2] L. Castelli Aleardi, O. Devillers and Abdelkrim Mebarki. Catalog based representation of 2D triangulations. *Internat. J. Comput. Geom. Appl.*, 21(4): 393-402, 2011. — /

[J1] L. Castelli Aleardi, O. Devillers and Gilles Schaeffer. Succinct representations of planar maps, *Theoretical Computer Science (special issue in honor of F. Preparata)*, 408:174-187, 2008. — /

International conferences

[C4] L. Castelli Aleardi, O. Devillers and G. Schaeffer. Optimal succinct representations of planar maps, in *Proc. of the 22nd annual ACM Symposium on Computational Geometry (SoCG 2006)*, p. 309-318. — /

[C3] L. Castelli Aleardi, O. Devillers et A. Mebarki. 2D Triangulation representation using stable catalogs, in *Proc. of the 18th Canadian Conference on Computational Geometry (CCCG 2006)*, p. 71-74. — /

[C2] L. Castelli Aleardi, O. Devillers et G. Schaeffer. Succinct representation of triangulations with a boundary, in *Proc. of the 9th Workshop on Algorithms and Data Structures (WADS 2005)*, LNCS, vol. 3608, p. 134-145. — /

[C1] L. Castelli Aleardi, O. Devillers and G. Schaeffer. Dynamic updates of succinct triangulations, in *Proc. of the 17th Canadian Conference on Computational Geometry (CCCG 2005)*, p. 135-138. — /

ACKNOWLEDGEMENTS

I extend my gratitude to my parents, Manlio Castelli and Maria Grazia Aleardi, and my family for their unwavering love and support. Their consistent encouragement bolstered me through the challenging and fulfilling phases of my academic journey.

I would like to acknowledge that the works presented in this collection are the result of collaborative efforts, symbolizing collective achievements. I appreciate all my co-authors and numerous colleagues for the enriching and enjoyable moments shared during our research endeavors and beyond.

Special thanks are due to the reviewers and the entire jury for their valuable constructive feedback and for generously committing their time to this process. I am sincerely grateful for their involvement in this academic endeavor.

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

Graphs and networks play a relevant role in modern science: graphs are clearly fundamental in computer science and related fields (e.g. discrete mathematics) and they provide powerful tools for abstraction in many other research domains, as they allow us to represent a collection of entities together with their interactions, as done in social networks and biological processes.

This manuscript focuses on the algorithmics of graphs which come with an extra topological and combinatorial structure, given by an embedding in the plane or on a surface of bounded genus (they are also referred to as *maps*). These graphs are among the most important objects in several application domains, as they correspond to the combinatorial structure underlying 3D meshes (one of the the most used representations for describing a discrete approximation of a surface). The wide diffusion and the increasing complexity of planar and surface meshes motivated a large number of works, in the last two decades, addressing the problems of efficiently processing, representing and visualizing such objects. Consequently, my research is at the frontier between several domains, ranging from computational geometry and geometry processing to combinatorics, graph drawing and data structures.

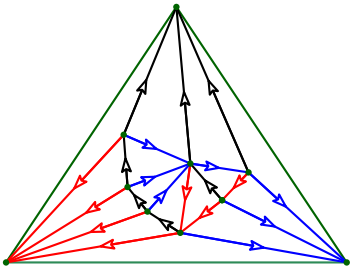


Figure 1.1: A plane triangulation endowed with a Schnyder wood.

COMBINATORIAL STRUCTURES FOR PLANAR GRAPHS. Planar graphs exhibit many important structural properties: one of the nicest characterizations is the one provided by *Schnyder woods*, a deep combinatorial structure which is able to finely capture the notion of planarity of a graph. Schnyder woods have been introduced by W. Schnyder to define a new planarity criterion in terms of poset dimensions [Sch89], as well as a very elegant and simple straight-line drawing algorithm [Sch90]. The most classical formulation is for the family of plane triangulations, yielding the following striking property: the internal edges of a triangulation can be partitioned into three trees that span all inner vertices and are rooted respectively at each of the three vertices incident to the outer face (as illustrated in Fig. 1.1). This nice global characterization in terms of spanning trees motivated a large number of generalizations and applications in several domains such as graph drawing, graph encoding, random sampling, succinct representations and plane spanners [Sch90; Chu+98; HKL99; BTV99; Fel01; BGH03; PS06; FPS05; Bon+06; Bar+12; CDS08; Bon+10b; Bon+10a; Dha10].

1.2 MY CONTRIBUTIONS

Previous existing works on Schnyder woods and related structures (e.g. canonical orderings) focused mainly on their applications to graph drawing and encoding, and on their extensions [Kan96; BTV99; Fel01] to other classes of planar graphs (e.g. 3-connected plane graphs).

In this manuscript we focus on triangulations and, more precisely, we will consider triangulations embedded on surfaces of arbitrary genus.

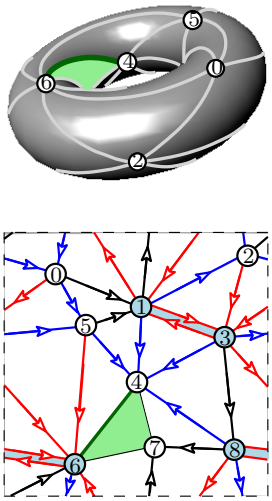


Figure 1.2: Example of g-Schnyder wood in the toroidal case.

1.2.1 *Schnyder woods and canonical orderings in higher genus (Chapter 3)*

As often occurs when dealing with combinatorial objects, the most difficult question involving a generalization is to find the proper definitions of such objects, and to prove their existence. In a work in collaboration with E. Fusy and T. Lewiner [CAFL09] we proposed the first generalization of Schnyder woods for the case of triangulations of higher arbitrary genus. One main contribution of this work is to show how to extend the computation of a canonical ordering (based on an incremental shelling algorithm) to the case of triangulations of genus g. As a by-product, we define a g-Schnyder wood which generalizes the planar well known structure preserving the global spanning properties: roughly speaking, in the higher genus case is possible to obtain an edge orientation where most of vertices still respect the local original Schnyder rule, while a small set of at most $4g$ vertices may have more than 3 outgoing incident edges (see Fig. 1.2 for an illustration).

Since then, the study of orientations for graphs on surfaces has attracted more attention [GL14; AGK16; DGL17; KGL19; Sua21]. In particular, different generalizations of Schnyder woods and canonical orderings have been proposed in the toroidal case for graph drawing purposes [GL14; CDF18b].

1.2.2 *Graph encoding and practical compact data structures (Chapter 4)*

A general way to encode the combinatorial information of a graph embedded on a surface consists in choosing a (vertex) spanning tree T and to encode the tree T along with the set edges not lying in the tree. This approach, combined with Schnyder woods and canonical orderings, leads in the planar case to efficient linear-time encoding schemes. For instance, for the class of plane triangulations having n vertices, the Schnyder tree decomposition allow us to encode a triangulation [HKL99] with at most $4n$ bits using multiple parenthesis words: this encoding is very elegant and simple to implement, achieving a compression rate which is quite close to the information theory optimal bound of $3.24n$ bits.

ENCODING HIGHER GENUS TRIANGULATIONS. The graph encoding problem was one of the main motivations for introducing our generalization of Schnyder woods [CAFL09]. A consequence of our definition of g-Schnyder woods is that the global spanning condition can be preserved: the spanning tree decomposition of Schnyder woods (of the planar case) translates into the existence of a partition of the edges into a cut-graph (a map embedded on a surface having one face) more two spanning maps of genus g. This tool allowed us to obtain a new encoding of genus g triangulations of size at most $4n + (g \log n)$ bits, which matches the bound of the Edgebreaker compression scheme [Ros99], being quite closed to the theoretical minimum bound [1] of $3.24n + \Omega(g \log n)$ bits.

SPACE-EFFICIENT DATA STRUCTURES. We also consider the graph encoding problem in a different setting: in a work in collaboration with O. Devillers [CD18] we address the problem of designing practical compact data structures, allowing us to efficiently perform local navigation and answer adjacency queries, while requiring a small amount of memory resources. More precisely, we show how to exploit the spanning tree decomposition provided

1 As we have shown in [CFL10], this bound is achieved combining efficient planarizing strategies with our optimal encoding of planar triangulations with multiple boundaries (see Section 4.3).

by Schnyder woods in order to design new parsimonious data structures for planar triangulations: when using (maximal) Schnyder woods, it is possible to represent a triangulation of size n with at most $4n$ references, supporting navigation in worst case $O(1)$ time.

Moreover, a nice feature of our data structure is that it can be decoded in a streamable way (without memory overhead) directly from the compressed format for triangulations mentioned above (combining Schnyder woods and parenthesis words): this overcomes a common limitation of previous compact data structures, which require an intermediate (non-compact) mesh representation for running the pre-processing construction phase.

I have implemented this solution and performed tests on large triangle meshes having tens of millions of vertices, and its practical interest is confirmed by our experiments: it is possible to reduce considerably the amount of memory consumption (common mesh representations requires between $13n$ and $19n$ references) while supporting fast navigation in practice.

1.2.3 *Drawing graphs on surfaces (Chapter 5)*

The problem of computing crossing-free drawings of planar graphs has attracted a lot of attention from both the combinatorial and algorithmic point of views in the last four decades. Several solutions exist allowing us to obtain pleasing layouts achieving several aesthetic criteria, that help the readability and the exploration of the graph structure. The Koebe's circle packing approach, Tutte's spring embedding [Tut63], the shift-based FPP method [FPP90] or the face-counting principle by Schnyder [Sch90] are among the most widely studied algorithmic paradigms for computing crossing-free layouts in the planar case (the pictures in Fig. 1.3 show the Tutte and Schnyder drawings a of a random planar triangulation).

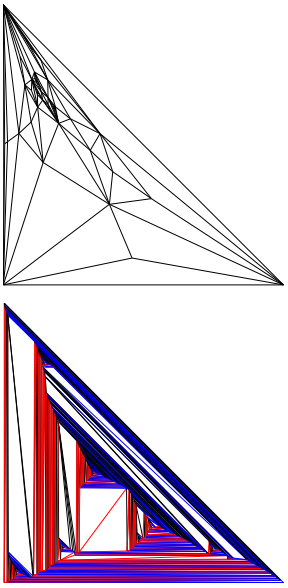


Figure 1.3: Planar straight-line drawings of a random planar triangulation with 100 vertices. (top) Tutte's barycentric embedding. (bottom) Schnyder drawing.

In this manuscript we focus on the problem of computing crossing-free layouts of graphs embedded on surfaces, which is very challenging and has been investigated less frequently.

PERIODIC DRAWINGS OF TOROIDAL MAPS. In a work [CDF12] in collaboration with E. Fusy and O. Devillers we observed that the shift-based FPP algorithm [FPP90] can be reformulated and adapted to deal with toroidal triangulations. In particular, it is be possible to extend the notion of canonical orderings to cylindrical triangulations: once a pair of parallel non-contractible cycles is chosen (a so-called tambourine), this leads to a new linear time (planar) drawing of a toroidal simple triangulation with n vertices, on a grid of size $O(n^{\frac{3}{2}} \times n)$. The main advantage of our approach, compared to previous works [DGK11], is the possibility to obtain an xy-periodic drawings in the flat torus, which leads to a periodic tiling of the plane (see Fig. 1.4 for an illustration). This approach is also suitable to deal with the more general case of essentially 3-connected toroidal maps [CDF18b]. To our knowledge, our algorithms for computing periodic drawings in the flat torus achieve the best known bounds on the grid resolution for both the triangular and 3-connected case. For instance, the face-counting principle [GL14] leads to a periodic drawing on a grid of size $O(n^2) \times O(n^2)$ (in the case of simple triangulations, without loops or multiple edges), and layouts of $O(n^3)$ area can be obtained with a variation of the shift algorithm [DGK11] (but not satisfying the periodicity constraints).

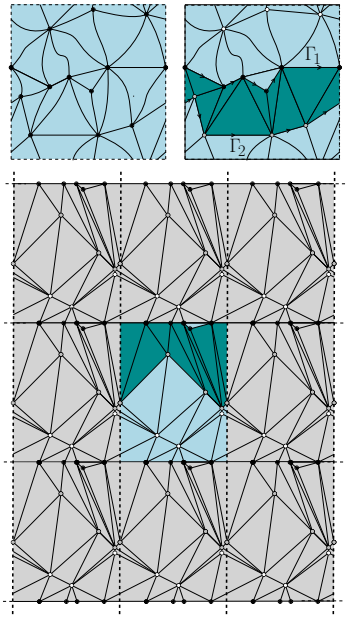


Figure 1.4: A periodic drawing of a toroidal triangulation in the flat torus (obtained with our modified version of the FPP algorithm).

In another work, with E. Fusy and A. Kostrygin [CFK14], we solved an interesting open question (mentioned in [Cha+12; DGK11]) which consists

in obtaining a planar grid drawing achieving simultaneously three criteria: polynomial area, and a straight-frame which is xy-periodic and whose boundary corresponds to a polygonal scheme of the graph.

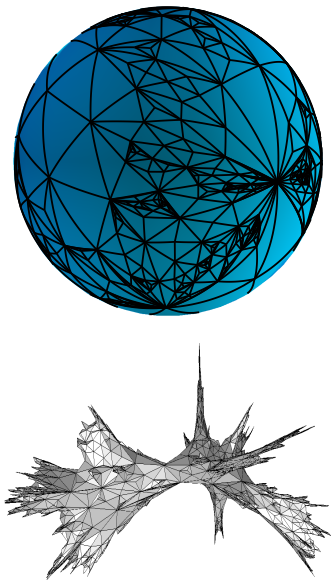


Figure 1.5: (bottom) 3D layout of a random triangulation with $5k$ faces. This layout is obtained running the FR spring embedder [FR91] using as initial layout a spherical drawing of the triangulation (top).

SPHERICAL DRAWINGS. The problem of drawing a planar graph with a prescribed boundary can also be applied to another important problem: the one of computing a geodesic spherical drawings of a genus 0 graph, which has many applications in geometric processing and computer graphics (also known as *spherical parameterization problem*). More precisely, we have devised an efficient and practical algorithm for computing a geodesic spherical drawing of a planar graph with polynomial resolution (Castelli Aleardi, Denis, and Fusy [CDF18a]). Quite interestingly, according to our experiments the computation of a spherical drawing can prove to be useful to provide good initial layouts for running 3D spring embedders (see Fig. 1.5): a good initial configuration does help to better and faster untangle the graph.

1.2.4 *Fast implementations and experimental results (Chapter 6)*

A significant part of some works [CD18; CDF18a; Cas19] has been devoted to the experimental evaluation of the performances of the combinatorial algorithms relying on Schnyder woods and canonical orderings. All experimental results and graph layouts presented in this manuscript have been obtained with my own Java implementations performing tests on several classes of graphs (having up to several millions of vertices): our experiments confirm the efficiency of combinatorial algorithms in terms of both running time and memory requirements. Beside the applications to graph drawing and graph encoding, it turns out the Schnyder woods have proven to be useful for other algorithmic problems involving planar graphs, such as the computation of cycle separators [Cas19].

1.3 OVERVIEW OF THE MANUSCRIPT

This document is intended to be self-contained as much as possible. The reader who is familiar with the main concepts of topological graph theory and combinatorics of planar maps may skip Chapter 2. Chapters 4 and 5 are independent and are conceived as short surveys providing the main recent advances concerning the problems of drawing and encoding graphs embedded on surfaces. For the sake of clarity and concision this document provides a high level and informal presentation of my works in the domains of graph drawing and graph encoding. I put strong emphasis on the advantages of Schnyder woods as deep tool for dealing with the algorithmics of graphs embedded on surfaces. My goal is to provide an intuition of the proofs underlying the main results, while skipping most technical details. Proofs are omitted or sometimes only sketched: the interested reader can refer to the published works for the complete proofs and missing details.

DISCLAIMER. The vast majority of results, both theoretical and experimental, are not new and appeared in journal or conference publications that I co-authored: this document reproduces large passages and some figures from these works. The only exception are the algorithmic proof for the computation of a toroidal Schnyder wood (described in Section 3.4.3), which can be seen as a by-product of our results on cylindrical maps, and the experimental results on Schnyder woods in genus $g \geqslant 1$ (Section 6.1).

2 PRELIMINARIES

2.1 GRAPHS AND MAPS (ON SURFACES)

For the sake of completeness, we provide a very brief description of the necessary concepts of topological graph theory we will need in this manuscript. The interested reader can find a more comprehensive and detailed discussion of these notions in [MT01].

2.1.1 *Definitions*

GRAPH EMBEDDINGS AND TOPOLOGICAL MAPS. A *graph* M *on a surface* is a graph $G = (V, E)$ *embedded* without edge-crossings on a closed orientable surface S (such a surface is specified by its genus g, i.e., the number of handles).

A celullar embedding *of a graph* G *is a crossing-free drawing of* G *on a surface* S *or, more precisely, a one-to-one map from* G *to* S *such that every vertex of* G *is map to a point on* S *and every edge* (u, v) *is represented as a simple path (curve) between the images of* u *and* v*. Moreover, the drawing is required to be crossing-free: the images of two distinct edges cannot cross (they meet only at their endpoints), and the images of edges cannot pass through the points images of vertices.*

If the components of $S \backslash G$ are homeomorphic to topological disks, then M is a so-called *(topological) map* (in this case the underlying graph G is necessarily connected). A subgraph $G' = (V', E')$ of G is called cellular if the components of $S \backslash G'$ are homeomorphic to topological disks, i.e., the graph G' equipped with the embedding inherited from G is a map.

Note that a map has more structure than a graph, since the edges around each vertex are in a certain cyclic order. In addition, a map has faces (the components of $M \backslash S$). By the Euler relation, the genus g of the surface on which M is embedded satisfies

$$2 - 2g = \chi(M) = |V| - |E| + |F|,$$

where $\chi(M)$ is the Euler characteristic of M, and V, E, and F are the sets of vertices, edges, and faces in M.

From the algorithmic point of view it is convenient to view each edge $e = \{u, v\} \in E$ as made of two *brins* [1], originating respectively at u and at v, the two brins meeting in the middle of e; the two brins of e are said to be *opposite* to each other. The *follower* of a brin h is the next brin after h in clockwise order (shortly cw) around the origin v of h. A *facial walk* is a cyclic sequence $(b_1, \ldots, b_k)$, where for $i \in [1..k]$, b_{i+1} (with the convention that $b_{k+1} = b_1$) is the opposite brin of the follower of b_i. A facial walk corresponds to a walk along the boundary of a face f of M in ccw order (i.e., with the interior of f on the left).

The face incident to a brin h is defined as the face on the left of h when one looks toward the origin of h. Note that to a brin h of M corresponds a *corner* of M, which is the pair $c = (h, h')$ where h' is the follower of h. The vertex incident to c is defined as the common origin of h and h', and the face f incident to c is defined as the face of M in the sector delimited by h and h' (so f coincides with the face incident to h).

COMBINATORIAL MAPS. Maps can also be defined in a combinatorial way. A *combinatorial map* M is a connected graph $G = (V, E)$ where one

1 Sometimes brins are also called *darts* in the literature. Brins are essentially equivalent to *half-edges*, which are at the core of popular data structures for representing graph embeddings [Ket99]: we refer to Section 2.3 for a more detailed presentation.

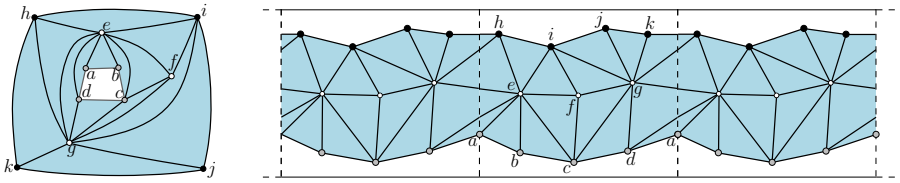


Figure 2.1: A cylindric triangulation with boundary faces $B_{inn} = \{a, b, c, d\}$ and $B_{ext} = \{h, i, j, k\}$. Left: annular representation. Right: x-periodic representation.

specifies a cyclic order for the set of brins (half-edges) around each vertex. One defines facial walks of a combinatorial map as above (note that the above definition of a facial walk as a certain cyclic sequence of brins does not need an embedding, it just requires the cyclic cw order of the brins around each vertex). One obtains from the combinatorial map a topological map by attaching a topological disk at each facial walk; and the genus g of the corresponding surface satisfies again $2 - 2g = |V| - |E| + |F|$, with F the number of topological disks (facial walks), which are the faces of the obtained topological map [MT01].

A graph is planar *if it admits an embedding on the sphere S^2 or, equivalently, if it can embedded into the plane. A* plane graph *is a graph endowed with its planar embedding.*

(PLANE) TRIANGULATIONS. In this document we will focus mainly on triangulations: more precisely, a *(simple) triangulation* $\mathcal{T}$ is a map with no loops nor multiple edges and with all faces of degree 3 (each face has 3 edges on its contour). If $\mathcal{T}$ is a *plane triangulation* the edges and vertices of incident to the outer face are called the *outer edges* and *outer vertices*. The other ones are called the *inner edges* and *inner vertices*. Most of times we deal with triangulations (or maps) which are *rooted*, having a distinguished root face.

In our drawings of plane triangulations the root face will coincide with the infinite outer face.

When the outer face is polygonal (degree greater than three) and all inner faces are triangular we have a so-called *quasi-triangulation*.

CYLINDRIC AND TOROIDAL MAPS. A *cylindric map* is a planar map with two marked faces B_{inn} and B_{ext} whose boundaries Γ_{inn} and Γ_{ext} are simple cycles (Γ_{inn} and Γ_{ext} might share vertices and edges). The faces B_{inn} and B_{ext} are respectively called the *inner boundary-face* and the *outer boundary-face* The other faces are called *inner faces*. If all inner faces are triangle then we have a so-called *cylindric triangulation*. Boundary vertices and edges are those belonging to Γ_{inn} (gray circles in Fig. 2.1) or Γ_{ext} (black circles in Fig. 2.1); the other ones are called *inner* vertices (white circles in Fig. 2.1) and edges. We also define a *chordal edge*, or *chord*, at Γ_{inn} as an edge not on Γ_{inn} but with its two ends on Γ_{inn}. Similarly a *chord* at Γ_{ext} is an edge not on Γ_{ext} but with its two ends on Γ_{ext}.

Through this work special attention will be paid to graphs embedded on the cylinder and on the torus (motivated by visualization purposes, in Chapter 5).

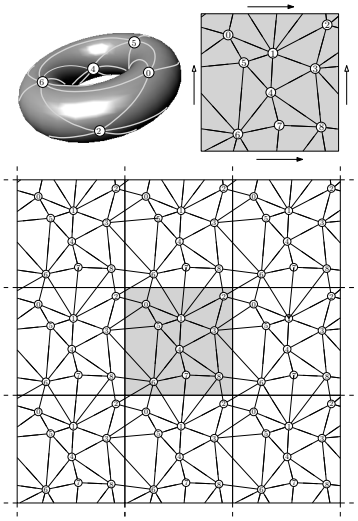


Figure 2.2: A toroidal triangulation and its periodic representation in the flat torus.

A *toroidal triangulation* is a map on the torus with only triangular faces (the map is called *simple* if it has no loop nor multiple edges).

PERIODIC REPRESENTATIONS OF CYLINDRIC AND TOROIDAL MAPS. We will often consider cylindric maps in the annular representation where B_{ext} is the outer face, as shown in Fig. 2.1(left). Sometimes we will draw cylindric (resp. toroidal) maps using the periodic representation in the flat cylinder (resp. torus).

For $w > 0$ and $h > 0$, the *flat cylinder* of width w and height h is the rectangle $[0, w] \times [0, h]$ where the vertical sides are identified (see right picture in Fig. 2.1). A point on this cylinder is located by two coordinates

$x \in \mathbb{R}/w\mathbb{Z}$ and $y \in [0, h]$. The *flat torus* of width w and height h is the rectangle $[0, w] \times [0, h]$ where both pairs of opposite sides are identified. A point on this torus is located by two coordinates $x \in \mathbb{R}/w\mathbb{Z}$ and $y \in \mathbb{R}/h\mathbb{Z}$. Figure 2.2 shows the drawing of a simple toroidal triangulation in the flat torus.

NON-CONTRACTIBLE CYCLES, CUT-GRAPHS, POLYGONAL SCHEMES. A subgraph $G' = (V', E')$ is *spanning* if $V' = V$. A *cut-graph* of M is a spanning cellular subgraph $G' = (V', E')$ with a unique face, i.e., $S \backslash G'$ is homeomorphic to a topological disk (the example of Fig. 2.3 shows a cut-graph of a toroidal triangulation). A non-contractible curve is a closed curve that cannot be continuously deformed to a point. Such a curve is called *proper* if it meets G only at vertices (not at edges); the *length* of a proper non-contractible curve is the number of vertices it meets. A *non-contractible cycle* (also called *fundamental cycle*) of G is a (simple) cycle of edges of G that forms a non-contractible curve (the length of such a cycle is its number of edges).

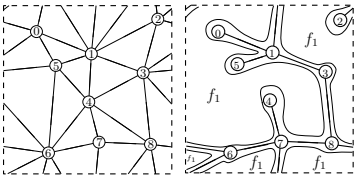


Figure 2.3: A toroidal triangulation and a cut-graph spanning all its vertices.

LCA: controler fundamental cycle.

In our setting, where we deal with a graph G embedded on a surface S, we consider a *canonical polygonal scheme* as a collection of $2g$ fundamental cycles, with a common vertex v, such that cutting S along these cycles lead to a topological disk.

ESSENTIALLY SIMPLE TRIANGULATIONS AND 3-CONNECTED MAPS. A cylindric map is called *simple* if it has no loops nor multiple edges, and is called *essentially simple* if it has no loops nor multiple edges in the periodic representation. Note that an essentially simple cylindric map might have 2-cycles and 1-cycles (loops), which have to be *non-contractible* (they have B_{inn} on one side and B_{ext} on the other side), and two loops cannot be incident to the same vertex.

A toroidal map is called *essentially simple* if its periodic representation in the plane is simple. A toroidal map is called 3-*connected* if it is 3-connected as a graph, and is called *essentially* 3-*connected* if its periodic representation[2] in the plane is 3-connected (see Fig. 2.4).

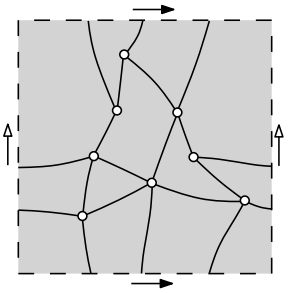


Figure 2.4: An essentially 3-connected toroidal map.

EDGE-WIDTH AND FACE-WIDTH. The *face-width* of G is the minimum of the lengths of the proper non-contractible curves for G. The *edge-width* of a toroidal map G is the minimum of the lengths of the non-contractible cycles of G. For a cylindric map, the *edge-distance* d between the two boundaries is the length of a shortest possible path starting from a vertex of Γ_{inn} and ending at a vertex of Γ_{ext} (possibly $d = 0$).

DUALITY. The *dual* of a (topological) map M is the map M^* on the same surface defined as follows: M^* has a vertex in each face of M, and each edge e of M gives rise to a *dual edge* e^* in M^*, which connects the vertices of M^* corresponding to the faces of M sharing e. Note that the adjacencies between the vertices of M^* correspond to the adjacencies between the faces of M. Duality for edges can be refined into duality for brins: the dual of a brin h of an edge e is the brin of e^* originating from the face incident to h (the face on the left of h when looking toward the origin of h). Note that the dual of the dual of a brin h is the opposite brin of h.

2 The periodic representation (also called universal cover) is obtained by gluing an infinite number of translated copies of our drawing, assembling the right (resp. upper) side of one copy with the left (resp. lower) side of another. Then a non contractible loop become an edge linking two different translated copies of the same point.

2.1.2 *Planarizing graphs on surfaces*

A possible solution to deal with algorithmic problems involving higher genus graphs consists in performing a planarization of the surface. Actually, given a triangulation $\mathcal{T}$ with n vertices on a surface S of genus g, one can compute a cut-graph or a collection of $2g$ non-trivial cycles, whose removal makes S a topological disk (possibly with boundaries). For example some work makes it possible to compute polygonal schemas in time $O(gn)$ for a triangulated orientable manifold [Laz+01; VY90]. Nevertheless a planarization approach would not be always best suited in some cases: from the combinatorial point of view this would imply to deal with boundaries of arbitrary size (arising from the planarization procedure), as non-trivial cycles can be of size $\Omega(\sqrt{n})$, and cut-graphs have size $O(gn)$. Moreover, from the algorithmic complexity point of view, the most efficient procedures for computing small non-trivial cycles [CM07; Kut06] require more than linear time, the best known bound being currently of $O(n \log n)$ time.

Planarizing toroidal graphs

When dealing with toroidal graphs the planarization problem become easier, especially when there no loops and mo multiple edges: depending on the intended purpose one can cut along one or more non-contractible cycles, leading for instance to cylindric maps or pairs of subgraphs which are planar.

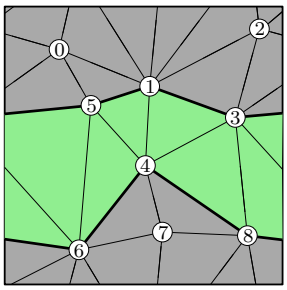


Figure 2.5: Cutting along two parallel non-contractible cycles: the toroidal triangulation is decomposed into a tambourine (green faces) and a cylindric map (gray faces).

TAMBOURINE. Let M be a toroidal map, and let Γ_1, Γ_2 be a pair of homotopic non-contractible cycles of M that are oriented in the same direction. The pair Γ_1, Γ_2 is called a *tambourine* if the area A between Γ_1 and Γ_2 — on the right of Γ_1 and on the left of Γ_2— is a "ribbon of faces", i.e., A is face-connected and the set of edges dual to edges in $A \backslash \{\Gamma_1, \Gamma_2\}$ forms a non-contractible cycle homotopic to Γ_1 and Γ_2. The drawing of Fig. 2.5 gives an example of such a configuration. Note that Γ_1 and Γ_2 might share vertices and edges. In its master's thesis Arnaud Labourel (see also [BGL05]) shows that any toroidal triangulation admits a tambourine decomposition, more precisely

Proposition 2.1 ([BGL05]) *Let $\mathcal{T}$ be a simple toroidal triangulation and let Γ be a non-contractible cycle of $\mathcal{T}$. Then there is a tambourine Γ_1, Γ_2 of $\mathcal{T}$ whose two cycles are homotopic to Γ. Moreover one can compute such a tambourine in linear time.*

It turns out that this statement holds more generally for essentially 3-connected toroidal maps, as we proved in [CDF18b].

CUTTING A TOROIDAL TRIANGULATION ALONG THREE CYCLES. Another way of planarizing a toroidal triangulation $\mathcal{T}$ consists in cutting along three non-contractible and non-homotopic cycles (whose intersection is a single vertex), which leads to a decomposition of $\mathcal{T}$ into two plane quasi-triangulations [3]. See Fig. 2.6 for an example. The correctness of this approach relies on the following fact (cited in [GL14] as unpublished result):

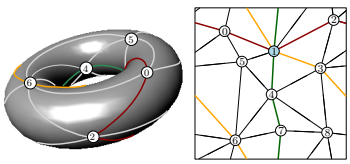


Figure 2.6: Three non-contractible and non-homotopic cycles crossing at vertex 1

3 This planarization approach has been used to prove the existence of Schnyder wood for simple toroidal triangulations: the solution proposed by Gonçalves and Lévêque [GL14] will be sketched in Section 3.4

Proposition 2.2 (Fijavz) *Any simple toroidal triangulation $\mathcal{T}$ contains three non-contractible and non-homotopic cycles Γ_1, Γ_2 and Γ_3 whose intersection is a single vertex (they are pairwise disjoint otherwise).*

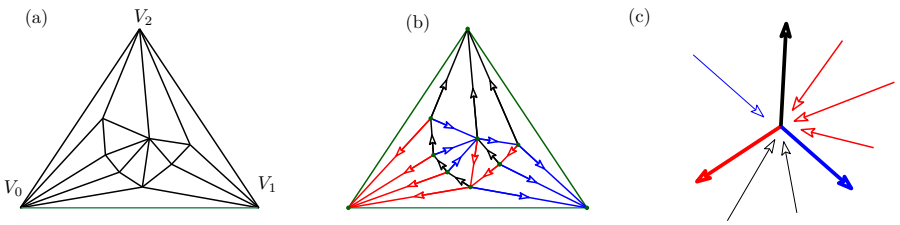


Figure 2.7: A rooted plane triangulation with 10 vertices, with root face (V_0, V_1, V_2) (a), endowed with a Schnyder wood (b). Picture (c) shows the local Schnyder condition around inner vertices.

2.2 SCHNYDER WOODS (AND CANONICAL ORDERINGS)

There are several equivalent formulations of Schnyder woods, either in terms of *angle labeling* (Schnyder labeling) or *edge coloring and orientation* or in terms of orientations with prescribed out-degrees. In this manuscript we focus on the formulation given in terms of edge orientations for the family of plane triangulations.

An α-orientation *of a graph* $G = (V, E)$ *(where* $\alpha : V \longrightarrow \mathbb{N}$*) is a choice of a direction for each edge, such that for each vertex* $v \in V$*, there are exactly* $\alpha(v)$ *outgoing edges incident to* v*.*

Schnyder woods, and more generally α-orientations, received a great deal of attention: from the combinatorial point of view, the set of Schnyder woods of a fixed triangulation has an interesting lattice structure [Bre00; Bon02; Fel04; DFM01; Men94], and the nice characterization in terms of spanning trees motivated a large number of applications in graph drawing, graph coding and random sampling [Sch90; Chu+98; HKL99; BTV99; Fel01; BGH03; Fus07; PS06; FPS05; Bon+06; Bar+12].

2.2.1 *Schnyder woods and canonical orderings for plane triangulations: definition*

In this manuscript the labels are also referred to as colors*. The edges of color* 0 *will be drawn as red segments or curves, while we will use blue and black segments for edges of color* 1 *and* 2 *respectively.*

We recall here the definition of Schnyder woods for plane triangulations given in terms of edge colorings and orientations (illustrated in Figure 2.7).

Definition 2.3 (Schnyder [Sch90]) *Let* T *be a plane triangulation, and denote by* V_0, V_1, V_2 *the outer vertices in counterclockwise (*ccw*) order around the outer face. A* Schnyder wood *of* T *is an orientation and labeling, with labels in* $\{0, 1, 2\}$*, of the inner edges of* T *so as to satisfy the following conditions:*

A Schnyder wood of a plane triangulation T *defines a so-called* 3-orientation *of* T*: all inner vertices have exactly* 3 *outgoing edges (and outer vertices have no outgoing edges).*

- ***root-face condition:*** *for* $i \in \{0, 1, 2\}$*, the inner edges incident to the outer vertex* v_i *are all ingoing of color* i*.*
- ***local condition for inner vertices:*** *For each inner vertex* v*, the edges incident to* v *in counterclockwise (*ccw*) order are: one outgoing edge colored* 2*, zero or more incoming edges colored* 1*, one outgoing edge colored* 0*, zero or more incoming edges colored* 2*, one outgoing edge colored* 1*, and zero or more incoming edges colored* 0*, which we write concisely as*

$$(\mathrm{Seq}(\mathrm{In}\,1), \mathrm{Out}\,0, \mathrm{Seq}(\mathrm{In}\,2), \mathrm{Out}\,1, \mathrm{Seq}(\mathrm{In}\,0), \mathrm{Out}\,2).$$

While the definition is given in terms of local conditions, the main structural property, as stated in Fact 2.4, is more global, namely a partition of the inner edges into 3 trees (please refer [Bre00] for more details).

Fact 2.4 (Schnyder [Sch90]) *Each plane triangulation* T *admits a Schnyder wood. Given a Schnyder wood on* T*, the three directed graphs* T_0*,* T_1*,* T_2 *induced by the edges of color* 0*,* 1*,* 2 *are trees that span all inner vertices and are naturally rooted at* V_0*,* V_1*, and* V_2*, respectively.*

Another important consequence of the local Schnyder condition is that one get a partition of the inner faces of a triangulation following the three paths in T_0, T_1, T_2 starting at an inner vertex v. More precisely we have:

Fact 2.5 (Schnyder [Sch90]) *Let us consider a plane triangulation $\mathcal{T}$ endowed with a Schnyder wood, and the three paths $\Pi_i(v)$ (for $i \in \{0,1,2\}$) in T_i from an inner vertex v to the vertex V_i on the root face. Then the paths Π_0, Π_1 and Π_2 are disjoint (the only share vertex v) and provide a partition of the $2n-5$ inner faces of $\mathcal{T}$ into three regions $R_0(v)$, $R_1(v)$ and $R_2(v)$ (as illustrated in Fig. 2.8).*

This simple fact is crucial for designing efficient grid drawing algorithms, as sketched in Chapter 5.

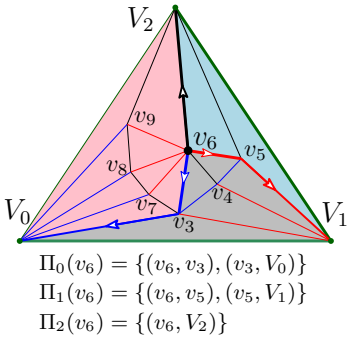


Figure 2.8: This picture illustrates the partition into three regions $R_i(v_6)$ defined by the three paths $\Pi_i(v_6)$ emanating from vertex v_6.

CANONICAL ORDERINGS. Another important combinatorial structure, closely related to Schnyder woods, is given in terms of a shelling order on the vertices of a graph. For the case of plane triangulations, *canonical orderings* can be defined in the following way (see Fig. 2.9):

Definition 2.6 ([FPP90]) *Let $\mathcal{T}$ be a plane triangulation, whose vertices on the outer (root) face are denoted V_0, V_1, V_2. An ordering $\pi = \{v_1, v_2, \ldots, v_n\}$ of the n vertices of $\mathcal{T}$ is called a* canonical ordering *if the subgraphs G_k ($3 \leqslant k \leqslant n$) induced by the vertices $v_1, \ldots, v_k$ satisfy the following conditions (where we denote by B_k the cycle bounding the outer face of G_k):*

- *G_k is 2-connected and internally triangulated, and $G_n = \mathcal{T}$;*
- *v_1 and v_2 belong to the outer face (V_0, V_1, V_2);*
- *for each $k \geqslant 3$ the vertex v_k is on the B_k and its neighbors in G_{k-1} are consecutive on B_{k-1}.*

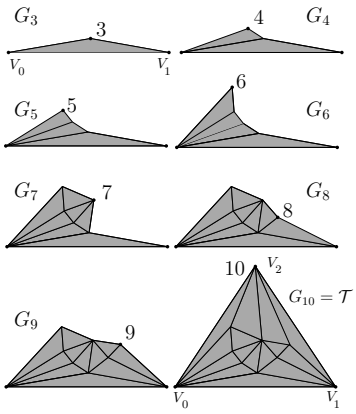


Figure 2.9: A sequence of graphs $\{G_k\}_{k \leqslant n}$ corresponding to a canonical ordering of a triangulation $\mathcal{T}$ ($n = 10$).

It is worth noting that canonical orderings and their generalizations are a very versatile tool for dealing with the algorithmics of planar graphs, from graph drawing to graph encoding, as we will in the next chapters.

2.2.2 *Computing Schnyder woods (and canonical orderings) in the plane*

In view of the generalization of Schnyder woods to higher genus, for the sake of completeness we quickly review two procedures for computing Schnyder woods (we mainly focus on the case of plane triangulations): one approach is based on a *vertex shelling procedure*, while the second makes use of *edge contractions*. Both approaches have been generalized independently, leading to different definitions of Schnyder woods for higher genus surfaces [CAFL09; GL14]: Chapter 3 will be devoted to illustrate these results.

Vertex shelling.

We first review a well-known linear time algorithm designed for computing a Schnyder wood of a plane triangulation, following the presentation by Brehm [Bre00]. It is convenient here to consider a plane triangulation as embedded on the sphere S^2, with a marked face that plays the role of the outer face. The procedure consists in growing a region C, called the *conquered region*, delimited by a simple cycle B (B is considered as part of C) [4]. Initially C consists of the root-face (as well as its incident edges and vertices). A *chordal*

4 In the figures, the faces of $\mathcal{T} \setminus C$ are shaded.

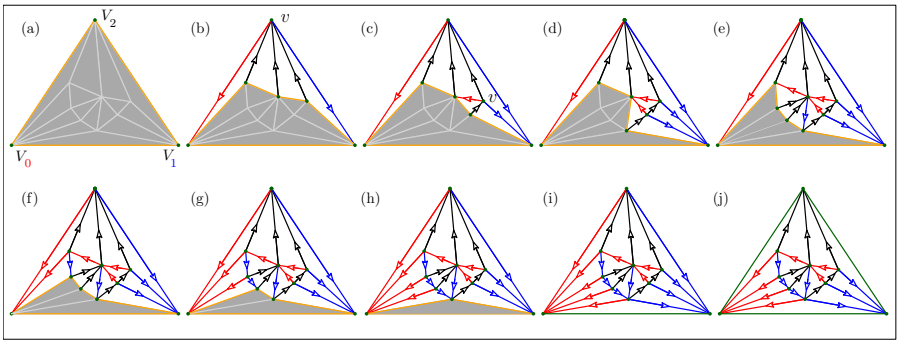


Figure 2.12: This figure illustrates the linear-time shelling procedure that computes a (maximal) Schnyder wood of a plane triangulation. The algorithm consists in performing $n-2$ vertex conquests, while maintaining a simple boundary cycle B (orange edges) enclosing the region that remains to be visited (gray faces). At the beginning (a) the boundary cycle coincides with the outer cycle (V_0, V_1, V_2). In order to ensure that B remains always simple (and the gray region simply connected), we avoid removing vertices on B that are incident to chordal edges (connecting two vertices on B). To get the maximal Schnyder wood it suffices to perform, at each step, the conquest of the free vertex closest to V_1.

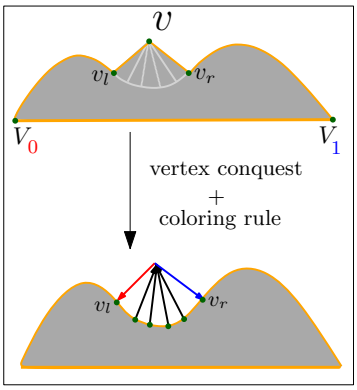


Figure 2.10: Vertex conquest

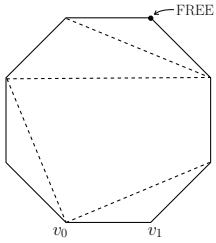


Figure 2.11: In a planar chord diagram with a root-edge $e = \{V_0, V_1\}$, there must be a vertex v not incident to e nor to any chord.

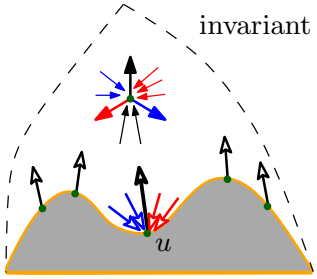


Figure 2.13: Invariant

edge is defined as an edge not in C but with its two extremities on B. A *free vertex* is a vertex of $B \setminus \{V_0, V_1\}$ with no incident chordal edges. One defines the *conquest* of such a vertex v as the operation of transferring to C all faces incident to v, as well as the edges and vertices incident to these faces; the boundary B of C is easily verified to remain a simple cycle. Associated with a conquest is a simple rule (depicted in Figure 2.10) to color and orient the edges incident to v in the exterior region. Let v_r be the right neighbor and v_l the left neighbor of v on B, looking toward $T \setminus C$ (in the figures, toward the shaded area). Orient outward of v the two edges (v, v_r) and (v, v_l); assign color 0 to (v, v_r) and color 1 to (v, v_l). Orient toward v and color 2 all edges exterior to C incident to v (these edges are between (v, v_r) and (v, v_l) in ccw order around v).

The algorithm for computing a Schnyder wood of a plane triangulation with n vertices is a sequence of $n-2$ conquests of free vertices, together with the operations of coloring and orienting the incident edges (the initial conquest, always applied to the vertex V_2, is a bit special: the edges going to the right and left neighbors are not colored nor oriented, since these are outer edges).

The correctness and termination of the traversal algorithm described above is based on the following fundamental property illustrated in Figure 2.11. A planar chord diagram (i.e., a topological disk with chordal edges that do not cross each other) with root-edge $\{V_0, V_1\}$ always has on its boundary a vertex $v \notin \{V_0, V_1\}$ not incident to any chord, see for instance [Bre00] for a detailed proof.

One proves that the structure computed by the traversal algorithm is a Schnyder wood by considering some invariants (see Figure 2.13):

- the edges that are already colored and directed are the inner edges of $C \setminus B$.
- for each inner vertex v of $C \setminus B$, all edges incident to v are colored and directed in such a way that the Schnyder rule (Figure 2.7(c)) is satisfied;

- every inner vertex $v \in B$ has exactly one outgoing edge e in $C \backslash B$; and this edge has color 2. Let v_r be the right neighbor and v_l the left neighbor of v on B, looking toward $\mathcal{T} \backslash C$. Then all edges strictly between (v, v_r) and e in cw order around v are ingoing of color 1 and all edges strictly between e and (v, v_l) in cw order around v are ingoing of color 0.

These invariants are easily checked to be satisfied all along the procedure (see [Bre00] for a detailed presentation), which yields the following result:

Lemma 2.7 (Brehm [Bre00]) *Let us consider a plane triangulation $\mathcal{T}$ with root face (V_0, V_1, V_2). Then the algorithm described above computes a Schnyder wood of $\mathcal{T}$ and can be implemented to run in time $O(n)$.*

Finally, observe that the incremental algorithm described above actually computes a canonical ordering of the input triangulation: it suffices to consider the sequence of vertex shellings in reverse order.

MINIMAL (MAXIMAL) SCHNYDER WOOD. Note that a triangulation $\mathcal{T}$ can have many different Schnyder woods (as shown by Brehm [Bre00], the set of Schnyder woods of $\mathcal{T}$ forms a distributive lattice). Furthermore, the same Schnyder wood can be obtained from many different canonical orderings for the above-described traversal procedure.

As one can observe, at each step there are many choices among the possible free vertices on B, which possibly leads to many distinct Schnyder woods. Among all Schnyder woods, the minimal (resp. maximal) one plays an important role in many algorithmic and combinatorial problems. A fundamental property is that the minimal (resp. maximal) Schnyder wood has no ccw (resp. cw) oriented cycles of directed edges. The computation of the minimal (resp. maximal) Schnyder wood can be performed as before: at each step just perform the conquest of the free vertex v on B that is the closest to V_0 (resp. to V_1).

COMPUTING SCHNYDER WOODS VIA EDGE CONTRACTIONS. The algorithm originally proposed in [Sch90] for computing Schnyder woods was relying on a slightly different approach based on edge contractions.

An edge (u, v) is *contractible* if u and v have exactly two common neighbors (no separating triangle containing u and v): observe that contracting an edge (u, v) of a plane triangulation $\mathcal{T}$ leads to a new smaller triangulation $\mathcal{T}'$ (as illustrated in the sequence of left picture in Fig. 2.14).

A crucial observation is that for a plane triangulation $\mathcal{T}$ with root face (V_0, V_1, V_2) having at least 4 vertices it is possible to find a contractible inner edge (V_2, v): this leads to construct a sequence of edge contractions that incrementally simplify the original triangulation until we are left with 3 vertices.

Then a Schnyder wood of $\mathcal{T}$ can be obtained by performing a sequence of *expansions*, accordingly to the sequence of edge contractions (in reverse order): the arguments underlying the validity of this procedure can be found in [Bre00] and [NR04].

PRACTICAL IMPLEMENTATIONS. It is worth nothing that in the planar case the algorithm based on vertex shellings and the procedure based on edge contractions are essentially equivalent: for instance they both allows us to recover all Schnyder woods of a given plane triangulation. Nevertheless, although they lead both to linear running time, there is no doubt that

Among the applications of canonical orderings we would like to mention an elegant linear-time algorithm for recognizing and embedding maximal planar graphs [NSI04]: its simplicity relies on the fact that in the triangular case the computation of a canonical ordering only requires the graph structure, since prior knowledge on the embedding is not needed. To detect free vertices it is actually sufficient to maintain the set S_B of vertices having a neighbor already removed, and to chose a vertex with exactly two neighbors on S_B.

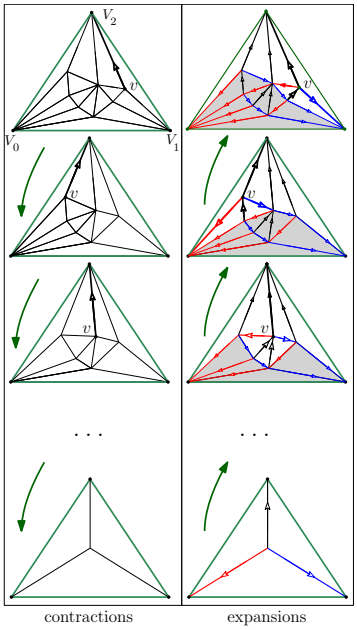


Figure 2.14: Computation of Schnyder woods via edge contractions.

As we will see in the Chapter 3 the non planar case is much more involved: the approach based on vertex shellings and the one using edge contractions lead to distinct generalizations of Schnyder woods, each preserving some, but not all, the structural properties of Schnyder woods.

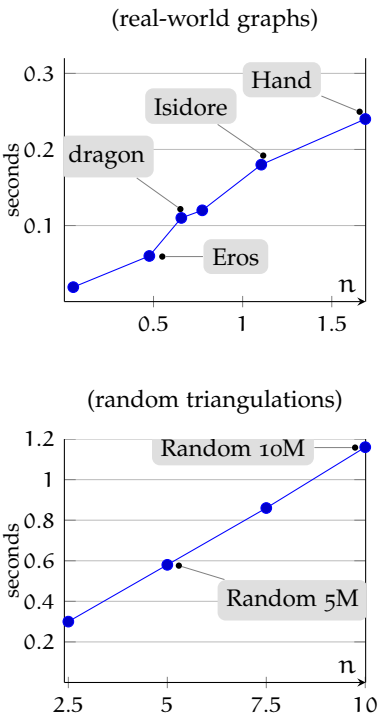


Figure 2.15: Runtime cost of the incremental shelling algorithm for the computation of Schnyder woods and canonical orderings. Timings are expressed in seconds as a function of the size (millions of vertices). We plot the average timing costs over 100 executions, obtained allocating 6GB of RAM for the JVM (results are taken from [CD18]).

```
class Halfedge{
  Halfedge  prev;
  Halfedge next;
  Halfedge opposite;
  Vertex v;
  Face f;
}
```

```
class Vertex{
  Halfedge halfedge;
  Point p;
}
class Face{
  Halfedge  e;
}
```

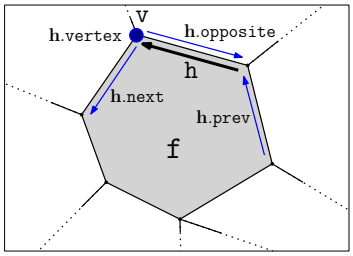


Figure 2.16: Illustration of the *half-edge* data structure: each half-edge stores three references towards its previous, next and opposite half-edges and the target vertex, while each vertex store a reference to an incoming incident half-edge. If one does not need to represent faces explicitly, this amounts to an overall storage of $2 \cdot 4e + n$ references for encoding a polygon mesh with e edges and n vertices. In the case of triangulations (assuming that the genus and the boundary size are small when compared to n), in its minimal form where we discard the `previous` half-edge, the storage cost of *half-edge* decreases to $3e + n = 18n + n$.

the vertex shelling approach is simpler to implement and much faster in practice. While performing vertex shellings only requires to maintain a simple boundary cycle, the computation of edge contractions requires to deal with a dynamic data structure for triangulations or to simulate the local updates in a non-trivial way.

Once an efficient representation of combinatorial maps is provided (we make use of a Java implementation of the *half-edge* data structure, described in next section), the computation of Schnyder woods and canonical orderings via vertex shellings is extremely fast. As observed in practice on large inputs, for both real-world and random triangulations, our `Java` implementation allows processing between 5.95M and 8.62M vertices per second (see Fig. 2.15).

2.3 IMPLEMENTATIONS AND EXPERIMENTAL SETTINGS

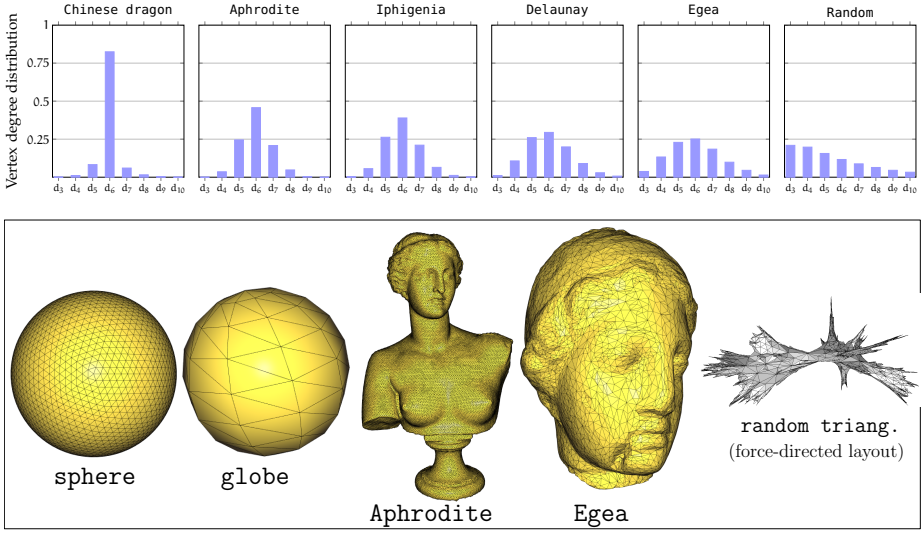
The relevance of surface maps for application domains such as computational geometry, geometry modelling and computer graphics relies on the fact that surface meshes are among the most common representations of 3D shapes. A *polyhedral 3D mesh* is a collection of polygonal faces that define a discrete approximation of a surface. Most of times [5] people focus on *manifold* triangle meshes (all faces having degree 3), for which every edge is bounding either one or two triangles, and where the faces incident to a same vertex define a closed or open fan.

The *connectivity* of triangle meshes describes incident relations between mesh elements and corresponds to the combinatorics of triangulations embedded on orientable surfaces (simple maps, with no loops nor multiple edges). Assuming that the genus and the number of boundary edges is negligible when compared to the number n of vertices, the number m of faces is roughly equal to $2n$.

COMMON MESH REPRESENTATIONS. Most of the time mesh representations are implemented in the explicit pointer-based form, where references (or pointers) are used to describe incidence relations between mesh elements and navigation is performed throughout address indirection. One example is the popular *half-edge data structure* that falls within this framework: its implementation [Ket99] provides the representation of combinatorial maps offered by the `CGAL` library. As illustrated by the example in Fig. 2.16 each edge e is represented as a pair of half-edges (with opposite directions), each associated to the two faces sharing e. In order to represent the combinatorics of the mesh each half-edge h stores some information, namely: a reference to the target vertex, a reference to the opposite half-edge together with two references to the half-edges that follow and precede h in ccw order in the face associated to h. Each vertex stores a reference to an incoming incident edge and, when needed, faces can be represented storing a reference to an incident half-edge.

Sometimes, when dealing with triangular meshes, an alternative solution can be preferred. For instance, face-based representations [Boi+02] support navigational operators equivalent to the ones offered by the half-edge data structure while using smaller memory resources: one only stores 6 references per triangle (3 references for incident vertices, and 3 references for the three neighboring faces), plus one reference per vertex (describing the map

5 For instance, more than 70% of the geometric datasets made available by the `aim@shape` repository are 3D polygonal meshes (about 84% of which are manifold).



Real-world 3D meshes	vertices	faces	d_6	diam	diameter normalized	Synthetic and random graphs	vertices	faces	d_6	diam	diameter normalized
Egea	8268	16K	0.25	59	0.64	Stacked	88576	177K	0.222	8	0.026
Bunny	26002	52K	0.50	142	0.88	Globe	160002	320K	0.999	383	0.946
Aphrodite	46096	92K	0.45	229	1.06	Sphere	163842	327K	0.994	400	0.999
Iphigenia	49922	99K	0.38	200	0.89	Delaunay	100003	200K	0.294	138	0.43
Chinese Dragon	655980	1.3M	0.824	601	0.74	Random	100002	200K	0.116	78	0.24

Table 1: The table above reports the statistics for some of the real-world and synthetic graphs tested in this work: for large graphs ($n \geqslant 50k$) the values of the diameter are approximated. Last column reports the diameter normalized with respect to the number of vertices. The top charts report some vertex degree distributions: most regular graphs appear leftmost.

from vertices to faces). According to Euler formula, this leads to a storage cost of 13 *references per vertex* (rpv).

Both edge-based and face-based mesh data structures fully support local navigation allowing us to efficiently implement most geometry processing algorithms: the code provided in Fig. 2.17 illustrates how to turn around a vertex in order to compute its degree.

```
Degree(v) {
    // Enumerating edges incident to v
    e = v.halfedge;
    d = 0;
    h = e.next.opposite;
    while (h!=e){
        h = h.next.opposite;
        d = d + 1;
    }
    return d + 1;
}
```

Figure 2.17: Computation of the vertex degree using the half-edge data structure.

IMPLEMENTATIONS. All experimental results (and graph layouts) presented in this manuscript are obtained with our implementations of the algorithms and data structures developed in our works.[6] In particular, I have written Java implementations of mesh data structures (half-edge, winged-edge, triangle-based) and algorithms for graph drawing and graph encoding described in Chapters 4 and 5. All our tests are run on an HP EliteBook, equipped with 8GB of RAM and an Intel Core i7 2.60GHz, running under linux (Ubuntu 16.04) and with Java 1.8 64-bit.

DATASETS. We performed tests on a wide collection of graphs, whose sizes range from a few hundreds to millions of vertices. Our tests involve various kinds of datasets, including

6 Pure Java implementations of our algorithms and data structures, as well as input graphs, are available at `www.lix.polytechnique.fr/~amturing/software.html`.

- several real-world 3D meshes used in geometry processing: they are made available by the `aim@shape` and `Thingi10k` [ZJ16] repositories;
- planar random triangulations [7] generated with our implementation of the uniform random sampler by Poulalhon and Schaeffer [PS06];
- stack triangulations;
- Delaunay triangulations of random points in the plane;
- synthetic regular graphs with different shapes (`sphere`, `cylinder`, ...).

As done in geometric processing, we use the proportion of degree 6 vertices, denoted by d_6, to measure the regularity of a graph: d_6 is close to 1 for regular meshes, while is sometimes below 0.3 for irregular 3d models. The proportion of degree 6 vertices can be even much smaller for some special classes of graphs. For instance in the case of random planar triangulations the distribution of vertex degrees follows an exponential decay: as evaluated in [Lis99; Got03], the probability of having a vertex of degree i in a large random planar triangulation, as n goes to infinity, is $p_i = \frac{16(i-2)}{i}\binom{2i-2}{i-1}(\frac{3}{16})^i$ (as confirmed in our experiments, the proportion of degree 6 vertices in a large random triangulation is approximately 11.6%).

7 By random planar triangulation we mean here a simple triangulated planar map randomly chosen according to uniform distribution among all triangulations of size n. Observe that such random triangulations only satisfy topological planarity constraints, with no consideration of geometry: they are somehow pathological triangulations, being quite far for instance from Delaunay triangulations of random points in a planar domain.

3

SCHNYDER WOODS AND CANONICAL ORDERINGS FOR NON PLANAR GRAPHS

3.1 DEALING WITH HIGHER GENUS GRAPHS

It is worth noting that moving from the plane to higher genus surfaces, some of the beautiful properties of Schnyder woods might be lost. In principle a simple counting argument suggests that in genus 1 one could endow a triangulation with a 3-orientation (each vertex having outdegree 3). But unfortunately the correspondence between 3-orientations and Schnyder woods is no longer valid: the local Schnyder condition could be violated at some vertices, as illustrated in Fig. 3.1. And even when a 3-orientation yields an edge coloring satisfying the Schnyder local condition everywhere, there are examples for which the edges of the same color may form many disjoint circuits (see Fig. 3.2).

So, being conscious that something could be missed, we made some compromises: our definition [CAFL09] was aimed to preserve the global spanning condition even in higher genus. While we are not able to guarantee that the local Schnyder condition is satisfied everywhere, the number of exceptions is still a small constant when the genus is bounded. Our definition of Schnyder woods relies on an algorithmic approach, namely the correspondence between the computation of Schnyder woods and canonical orderings: this makes our construction quite simple to implement having the advantage to work for any genus g triangulated surface. It is worth noting that until very recently most existing works were only dealing with the genus 1 case [GL14; DGL17].

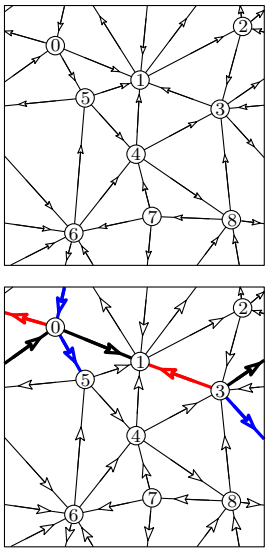


Figure 3.1: (top) A 3-orientation of a simple toroidal triangulation $\mathcal{T}$ with 9 vertices. (bottom) The corresponding edge coloring does not satisfy the local Schnyder condition everywhere. It suffices to propagate the edge coloration starting from vertex 0 and check that the incoming edges at vertex 1 do violate the local Schnyder condition (all incoming edges in the same sector should have the same color). Among all the 9360 existing 3-orientations of $\mathcal{T}$, only 2344 yield an edge coloring which is valid everywhere.

DISCLAIMER. The contributions of this chapter are originated from the collaborations with O. Devillers, É. Fusy and T. Lewiner [CAFL09; CDF18b].

PLANARIZING THE GRAPH. A possible intuitive solution to deal with Schnyder woods in higher genus would consist in performing a planarization of the surface. Nevertheless we must pay attention on how the planarization works, especially if one aims to preserve the local Schnyder condition (almost) everywhere: observe that we have to deal with boundaries of arbitrary size, since non-trivial cycles can be of size $\Omega(\sqrt{n})$ and cut-graphs are of size $O(gn)$, as already mentioned in Section 2.1.2.

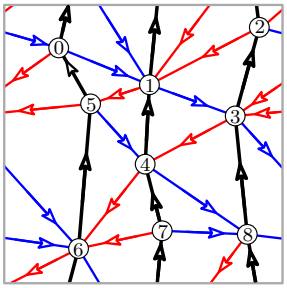


Figure 3.2: A 3-orientation whose corresponding edge coloring satisfies the local Schnyder condition at each vertex: the spanning condition is lost, since black edges define three disjoint cycles.

As far as we know the first attempt to makes use of Schnyder woods in the non planar case is due to Bonichon, Gavoille, and Labourel [BGL05]: their goal is to compute in linear time a partition of the edges into three edge-disjoint spanning trees plus at most 3 edges. Their solution, working in the genus 1 case relies on the computation of a tambourine (recall its definition given in Section 2.1.2): cutting along two parallel non contractible cycles we produce a tambourine plus a graph planar H that can thus be endowed with a Schnyder wood. It thus only remains to assign colors to the edges of the tambourine while avoiding cycles, which leads to the desired forest decomposition. Unfortunately, as pointed out by the authors, the local conditions of Schnyder woods are possibly not satisfied for a large number of vertices, because the size of the tambourine might be arbitrary large. We

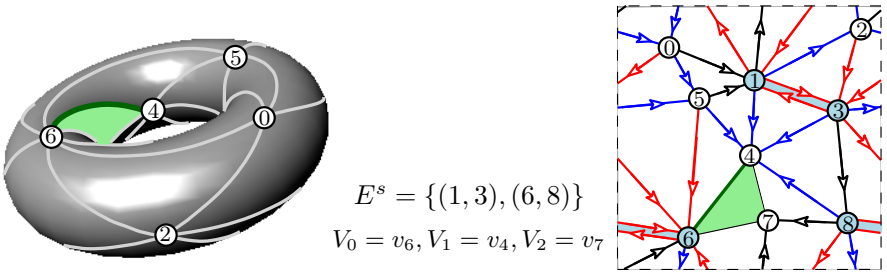


Figure 3.3: A toroidal triangulation endowed with a g-Schnyder wood (the root-face is green). For $g = 1$ there are two special edges, each of which is doubled and delimits a 2-sided face. In this example special edges are of same color.

point out that for toroidal graphs a different planarization approach has been later proved to be suitable in the triangular (simple) toroidal case: the elegant solution proposed by Gonçalves and Lévêque [GL14], satisfying the local Schnyder condition at each vertex, will be sketched in Section 3.4.

Let us observe that, for both methods above, it is not clear how to get a generalization to the case of genus $g \geqslant 2$.

3.2 g-SCHNYDER WOODS FOR TRIANGULATIONS OF ARBITRARY GENUS

3.2.1 *The definition*

Let us consider a genus g simple triangulation $\mathcal{T}$ with n vertices, having a root-face $f = (V_0, V_1, V_2)$ (the vertices are ordered according to a walk along f with the interior of f on the right). We propose the following definition (illustrated by Figures 3.3 and 3.4) of Schnyder woods that extends to arbitrary genus the notion known in the planar case (recall Definition 2.3):

Definition 3.1 (Castelli Aleardi, Fusy, and Lewiner [CAFL09]) *Let us denote by E^{in} the set of inner edges of $\mathcal{T}$. A* g-Schnyder wood *of $\mathcal{T}$ is a partition of E^{in} into a set of* normal edges *and a set E^s of* special edges *considered as fat, i.e., each special edge is doubled into two edges delimiting a face of degree 2, called a* special face*. In addition, each edge, a normal edge or one of the two edges of a special edge, is directed and has a label in $\{0,1,2\}$, so as to satisfy the following conditions:*

- ***root-face condition:*** *The outer vertex V_2 is incident to no special edges. All inner edges incident to V_2 are ingoing of color 2.*

  Let $k \geqslant 0$ be the number of special edges incident to V_0 (each of these special edges is doubled), and let $L = (e_1, f_1, e_2, f_2, \ldots, e_r, f_r)$ be the cyclic list of edges and faces incident to V_0 in ccw order (f_i is the face incident to V_0 between e_i and $e_{(i+1) \bmod r}$). A sector *of v is a maximal interval of L that does not contain a special face nor the root-face. Note that there are $k+1$ sectors, which are disjoint; the one containing the edge $\{V_0, V_1\}$ is called the* root-sector*.*

  Then, all inner edges in the root-sector are ingoing of color 0. In all the other k sectors, the edges in ccw order are of the form

  $$\mathrm{Seq}(\mathrm{In}\,1), \mathrm{Out}\,0, \mathrm{Seq}(\mathrm{In}\,2), \mathrm{Out}\,1, \mathrm{Seq}(\mathrm{In}\,0).$$

  The definitions of sectors and conditions are the same for V_1, except that all edges in the root-sector are ingoing of color 1.

- ***local condition for inner vertices:*** *Every inner vertex v has exactly one outgoing edge e of color 2. Let k be the number of special edges incident to v (each of these edges is doubled and delimits a special face), and let us consider the cyclic list $L = (e_1, f_1, e_2, f_2, \dots, e_r, f_r)$ of edges and faces incident to v in ccw order. A* sector *of v is a maximal interval of L that does not contain a special face nor the edge e. Note that there are $k+1$ sectors around v, which are disjoint (see Fig. 3.4 for an illustration).*

  Then, in each sector the edges in ccw order are of the form

$$\mathrm{Seq}(\mathrm{In}\,1), \mathrm{Out}\,0, \mathrm{Seq}(\mathrm{In}\,2), \mathrm{Out}\,1, \mathrm{Seq}(\mathrm{In}\,0).$$

- ***Cut-graph condition:*** *The graph T_2 formed by the edges of color 2 is a tree spanning all vertices except V_0 and V_1, and rooted at V_2, i.e., all edges of T_2 are directed toward V_2. The embedded subgraph G_2 formed by T_2 plus the two edges (V_0, V_2) and (V_1, V_2) plus the special edges (not considered as doubled here) is a cut-graph of $\mathcal{T}$, which is called the* cut-graph *of the Schnyder wood.*

(Note that the cut-graph condition forces the number of special edges to be $2g$.)

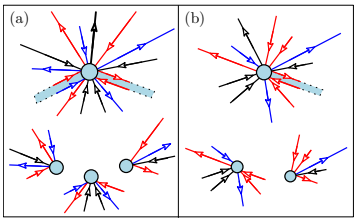


Figure 3.4: Local condition around special edges. (a) inner vertex incident to 2 special edges: there are 3 sectors delimited by the special edges and the outgoing edge of color 2. (b) inner vertex incident to one special edge: there are two sectors delimited by the outgoing edge of color 2.

Remark: *The last condition, stating that T_2 is a tree, is redundant in the planar case (it is implied by the local conditions) but not in higher genus: it is possible to find examples where all local conditions are satisfied but the edges of color 2 form many disjoint circuits.*

REMARKS. Note that if an inner vertex v is incident to no special edge, then there is a unique sector around v, which is formed by all edges incident to v except the outgoing one of color 2: the definition above implies that the edges around v the local Schnyder condition as in the planar case. Since at most $4g$ vertices are incident to special edges, our definition implies that in fixed genus, almost all inner vertices satisfy the same local condition as in the plane. In addition the vertices incident to special edges satisfy a local condition similar to the one in the planar case.

3.2.2 *Existence of g-Schnyder woods*

The difficulty of extending combinatorial constructions to higher genus is due the fact that some fundamental properties, such as the Jordan curve theorem, hold only in the planar case (genus 0). Following the approach suggested in [Lop+03; LLT03], based on Handlebody theory for surfaces, we design a new shelling algorithm for higher genus surfaces: as in the planar case, our strategy consists in conquering the whole graph incrementally.

We use an operator `conquer` similar to the conquest of a free vertex used in the planar case (described in Section 2.2.2), as well as two new operators—`split` and `merge`—designed to represent the handle attachments that are necessary in higher genus. Our traversal strategy is expressed in terms of subcomplexes: a short overview of the required terminology is provided below (for more details we refer to [CAFL09]).

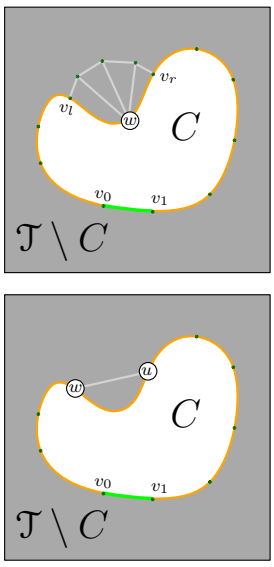


Figure 3.5: (top) Result of a `conquer` operation on a free corner incident to w and (bottom) a contractible chordal edge (w, u) (illustrated in the toroidal case).

SUBCOMPLEXES. Given a map M on a surface S, with V, E, and F the sets of vertices, edges, and faces of M, a *subcomplex* $C = (V', E', F')$ of M is given by subsets $V' \subset V$, $E' \subset E$, $F' \subset F$ such that the edges around any face of F' are in E' and the extremities of any edge in E' are in V'. Note that a connected subcomplex C of M naturally inherits from M the structure of a combinatorial map: it is then possible to define the notions of duality, Euler characteristic and facial walks for connected subcomplexes, as done for maps in Section 2.1.1. Facial walks are called *boundary walks* for C when they do not correspond to a facial walk of a face in F'. A boundary brin is a brin h in a boundary walk, and the corresponding *boundary corner* of C $b = (h, h')$ is the pair formed by h and the next brin h' in C in cw order

around the origin v of h. Note that a boundary corner of C is not a corner of M if there are brins $h_1, \dots, h_k$ of $M \backslash C$ in cw order strictly between h and h'.

The first operator, called `conquer`*, process free boundary corners, not modifying the topology of* C*.*

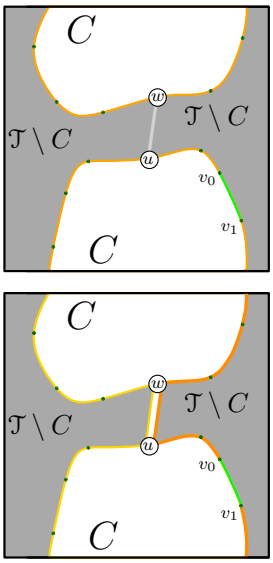


Figure 3.6: The result of a `split` operation on a split edge (w, u) (illustrated in the toroidal case). The boundary walk containing the extremities of (w, u) is split into two boundary walks (yellow and orange curves).

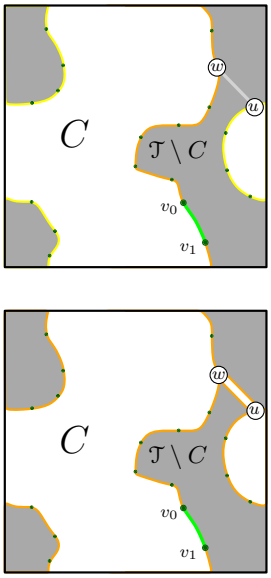


Figure 3.7: The result of a `merge` operation on a merge edge (w, u) (illustrated in the toroidal case). Informally a `merge` can be viewed as an operation that "adds a handle" to the subcomplex C.

CHORDAL EDGES AND FREE BOUNDARY CORNERS. A *chordal edge* is an edge of $\mathcal{T} \setminus C$ whose two brins h_1 and h_2 are exterior brins of some boundary corners b_1 and b_2. A boundary corner b of C is *free* if no exterior edge of b is a chordal edge. A chordal edge e for C is said to be *separating* if its dual edge e^* is a bridge of the dual complex D (a *bridge* is an edge whose removal disconnects the graph). Otherwise it is called non-separating.

HANDLE OPERATORS. Given b a free boundary corner of C, performing `conquer`(b) consists in adding to C all exterior faces of $\mathcal{T}$ incident to b, as well as the edges and vertices incident to these faces (the effect of `conquer`(b) is illustrated in Figure 3.9). Observe that D remains connected after the conquest and the number of boundary faces of C is unchanged, as well as the Euler characteristic: indeed, if the number of faces transferred to C is k, then the number of vertices transferred to C is $k-1$ and the number of edges transferred to C is $2k-1$.

Definition 3.2 *A* split edge *for* C *is a non-separating chordal edge* e *such that the two brins of* e *are incident to boundary corners in the same boundary face of* C*.*

Observe that given a split edge e its dual edge e^* is not a bridge but has the same boundary face (of D) on both sides. The second operation, called `split`, is acting on a split edge e as follows: double e into two parallel edges delimiting a face f of degree 2, and add the face f and the two edges representing e to C. Note that D remains connected since e^* is not a bridge. When doing the split operation, the boundary walk at the two extremities of e is split into two boundary walks. Therefore the number of boundary faces of C increases by 1. Note that the Euler characteristic $\chi(C)$ decreases by 1; indeed in C the number of vertices is unchanged, the number of edges increases by 2 (addition of the split edge, which is doubled) and the number of faces increases by 1 (addition of the special face). And the Euler characteristic of the map M associated with C is unchanged (when including the boundary faces, the number of faces both increases by 2, as the number of edges), hence the genus of M is also unchanged.

Definition 3.3 *A* merge edge *for* C *is a chordal edge having its two brins incident to boundary corners in distinct boundary faces of* C*.*

Observe that given a merge edge e, the faces of D on both sides of its dual e^* are distinct boundary faces, hence e^* cannot be a bridge of D, i.e., e is non-separating. We can now define the third operation, `merge`, related to a merge edge e: double e into two parallel edges delimiting a face f of degree 2, and add the face f and the two edges representing e to C. Note again that D remains connected since e^* is not a bridge. When doing a merge operation, the boundary faces at the two extremities of e are merged into a single boundary face, so that the number of boundary faces of C decreases by 1. Similarly as for a split operation, the Euler characteristic $\chi(C)$ decreases by 1 (addition of a doubled special edge and of one special face); and the Euler characteristic of the map M associated with C decreases by 2 (when including the boundary faces, the number of faces is unchanged, and the number of edges increases by 2), hence the genus of M increases by 1.

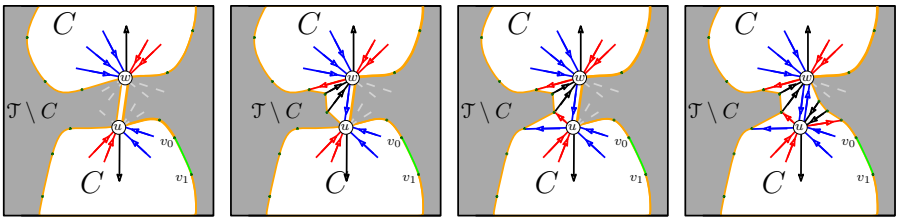


Figure 3.9: Illustration of the `colorient` operations. Any split (or merge) edge (u, w) can be directed in one or two directions, depending on the traversal order on its extremities (we denote by b_w a boundary corner incident to vertex w).

A SHELLING ALGORITHM IN ARBITRARY GENUS We can now describe an algorithm for computing a g-Schnyder wood that naturally extends to any genus the shelling procedure by Brehm and sketched in Section 2.2. Its the pseudo-code is given in Fig. 3.8 and Fig. 3.10 shows a complete execution of the algorithm in the toroidal case.

As in the planar case, the traversal is a greedy sequence of conquest operations, with here the important difference that these operations are interleaved with $2g$ merge/split operations. Another point is that, in higher genus, the region that is grown is more involved than in the planar case (recall that in the planar case, the grown region is delimited by a simple cycle). It also turns out that a vertex might appear several times on the boundary of the grown complex, therefore we have to use the refined notion of free boundary corner, instead of free vertex in the planar case (in the planar case, a vertex appears just once on the boundary of the grown region).

Let us now give the precise description of the traversal procedure on a rooted triangulation of genus g. As in the planar case, we grow a "region" C. Precisely, C is a connected subcomplex all along the traversal. Initially, C is the root-face $\{V_0, V_1, V_2\}$, together with the edges and vertices of that face; at the end, C is equal to $\mathcal{T}$. We make use of the operation `conquer(b)` (with b a free boundary corner of C) and of the associated `colorient` rule, similar to the operation for free vertices described in Section 2.2.2 (for the planar case). More precisely, given a free boundary corner b of C, the operation `colorient(b)` proceeds as follows: let v be the vertex incident to b, and let e, e' be the two edges delimiting b, with e' after e in cw order around v. Orient e and e' outward of v, giving color 1 to e and color 0 to e'. Orient all the exterior edges of b toward v and give color 2 to these edges (these edges are strictly between e and e' in cw order around v).

The algorithm, as illustrated by the pseudo-code in Fig. 3.8, performs a sequence of handle operations `split` and `merge` to deal with topology changes, interleaved with a sequence of `conquer+colorient` operations (corresponding to a vertex shelling of the subcomplex $\mathcal{T} \setminus C$).

Observe the subtlety that, for positive genus, the vertices incident to merge or split edges have several corners that are conquered, as illustrated in Figure 3.9. Precisely, for a vertex v incident to $k \geqslant 0$ merge/split edges, its conquest occurs $k+1$ times if v is an inner vertex and k times if $v \in \{V_0, V_1\}$.

Note also that, when the algorithm terminates, the number of merge edges must be g and the number of split edges must be g. Indeed, in the initial step, C has $k = 1$ boundary face and genus $g' = 0$, while (just before) the last step C has $k = 1$ boundary face and genus $g' = g$. Since the effect of each split is $\{k \leftarrow k+1, g' \leftarrow g'\}$ and the effect of each merge is $\{k \leftarrow k-1, g' \leftarrow g'+1\}$, there must be the same number of splits as merges

```
// T a simple triangulation of genus g
computeSchnyderAnyGenus(T) {
  Set C as the root-face f = (V0, V1, V2);
  while (C ≠ T) {
    find an update-candidate σ for C;
    If σ is a free boundary corner b
      conquer(b); colorient(b);
    If σ is a merge edge e = {u, w} for C
      merge(u, w);
    If σ is a split edge e = {u, w} for C
      split(u, w);
  }
}
```

Figure 3.8: Computation of g-Schnyder woods. An *update-candidate* for C is either a free boundary corner, or a split edge, or a merge edge. Note that the above algorithm performs conquests, merge operations and split operations in whichever order, i.e., with no priority on the 3 types of operations.

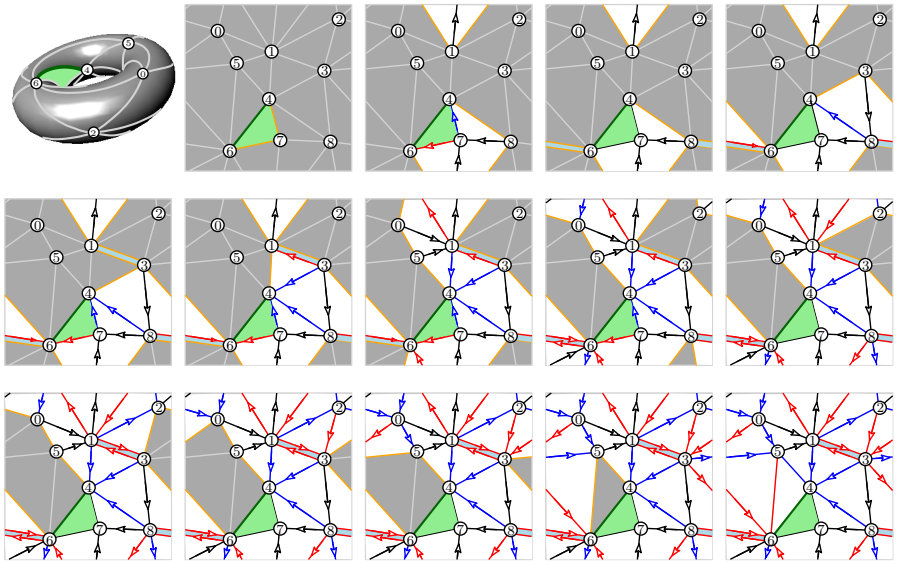


Figure 3.10: Execution of our traversal algorithm on a toroidal triangulation with 9 vertices. (a) The traversal starts with a conquest at the outer vertex $V_2 = v_7$. The area explored (white triangles) is homeomorphic to a disk. (b) Whenever there remain no free corners, `conquer` operations cannot be performed: thus it is possible to find a *split* edge (light blue edge). We then perform the conquest of a free corner (c) until we get stuck. We look for a *merge* edge: after the merge operation the region $\mathcal{T} \setminus C$ becomes a topological disk (d), the traversal can be completed with a sequence of `conquer` operations (e)-(m).

(for k to be the same finally as initially) and the number of merges must be g (for g' to increase from 0 to g).

The algorithm described above terminates computing an edge coloring and orientation that satisfies almost everywhere the local Schnyder condition: observe that the $2g$ merge and split edges processed by our shelling procedure are precisely the special edges involved in Definition 3.1. This leads to our main result:

Theorem 3.4 (Castelli Aleardi, Fusy, and Lewiner [CAFL09]) *Any simple triangulation $\mathcal{T}$ of genus g with n vertices admits a g-Schnyder wood, which can be computed in time $O((n+g)g)$.*

The termination relies on the fact that we can always find a candidate (a `conquer` or handle operation) at each step of the shelling procedure: in particular we look at candidates incident to the boundary face f_0 of C containing the root edge (V_0, V_1). Intuitively, if the vertex shelling phase get stuck at some step (no free boundary corners to conquer), then there exists a split or merge chordal edge e having one of its extremities on f_0, and the algorithm can perform a `split` or `merge` operation. Observe that such an edge e cannot be a separating chordal edge having its two extremities on f_0 otherwise it would enclose a planar region containing a free boundary corner of C (which would lead to a contradiction).

For the analysis of the runtime complexity, let us assume that we perform a maximal sequence of `conquer` operations, until the algorithm get stuck not finding any free boundary corner (this phase can be performed in linear time, using the suitable data structures, as in the planar case). Then we have to look for a split edge whose dual edge e^* is not a bridge of the dual complex: this requires linear time as well, as detecting all bridges in the dual

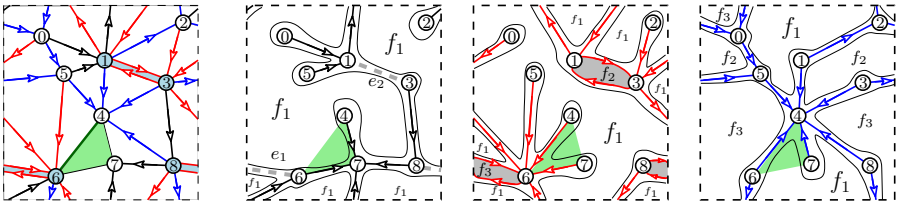


Figure 3.11: A toroidal triangulation endowed with a g-Schnyder wood. The edges of color 2 form a tree T_2, and the addition of the two special edges and the two outer edges incident to V_2 yields a cut-graph G_2. The edges of color 0 plus the two outer edges incident to V_0 form a spanning cellular subgraph G_0 with 3 faces (two faces are degenerated, having degree 2). Similarly, the edges of color 1 plus the two outer edges incident to V_1 form a spanning cellular subgraph G_1 with 3 faces.

can be done in $O(|E|)$ time using the depth-first search principles of [Tar74]. Observe that this phase needs to be performed at most $O(g)$ times.

The validity of the edge coloring computed by our traversal relies on some invariants on the colors and directions of the edges of a genus g triangulation $\mathcal{T}$ that remain satisfied all along the traversal and ensure that the computed structure is a g-Schnyder wood. The main difference between the planar and higher genus case involves the extremities of the special edges: observe that these edges can be considered as two parallel edges that delimit a face of degree 2. Special cases concern the sectors around V_0 and V_1 on the root face, which could be possibly be incident to some special edge (for instance this occurs in the example of Fig. 3.11).

A detailed proof of the correctness and runtime complexity of the algorithm above can be found in Sections 4.3 and 4.4 of [CAFL09].

3.2.3 *Spanning properties of g-Schnyder woods*

Finally, we point out that g-Schnyder woods give rise to decompositions into 3 spanning cellular subgraphs, one with one face and the two other ones with $1+2g$ faces. More precisely, let us consider a triangulation $\mathcal{T}$ of genus g endowed with a g-Schnyder wood computed by the algorithm ComputeSchnyderAnyGenus. Let us define G_2 as the graph formed by the edges of color 2, the two edges $\{V_1, V_2\}$ and $\{V_0, V_2\}$, and the 2g special edges (not considered as doubled). Let us define G_0 (resp. G_1) as the graph formed by the edges of color 0 (resp. color 1) plus the outer edges incident to V_0 (resp. incident to V_1): in this case we consider the special edges as doubled (thus $\mathcal{T}$ gets 2g additional degenerated faces of degree 2).

Proposition 3.5 ([CAFL09]) *The graphs G_0, G_1 and G_2 defined above satisfy the following properties:*

- *G_2 is a cut-graph of $\mathcal{T}$;*
- *G_0 and G_1 are spanning cellular subgraphs of $\mathcal{T}$ with $1+2g$ faces (where some of the faces might be degenerated, of degree 2).*

The properties of G_2 (cut-graph condition), and of G_0, G_1 stated above can be considered as extensions of the fundamental property of planar Schnyder woods [Sch89; Sch90]: in the planar case, for each color $i \in \{0,1,2\}$, the graph formed by the edges in color i plus the two outer edges incident to V_i is a spanning tree. Figure 3.11 shows an example in genus 1.

3.3 CANONICAL ORDERING FOR CYLINDRICAL SIMPLE TRIANGULATIONS

In this section we extend the notion of canonical ordering (classically studied on plane graphs) to cylindrical simple triangulations: we will explain in Section 5.3 how to make use of this tool for computing periodic drawings of toroidal graphs. An important observation must be made: we assume that the inner boundary Γ_{inn} of the input triangulation is chord-free.

We first defined the notion of cylindrical canonical orderings for cylindrical triangulations in [CDF12]: a generalization to essential 3-connected maps is given in [CDF18b].

Definition 3.6 (Castelli Aleardi, Devillers, and Fusy [CDF18b]) *Let G be a cylindrical simple triangulation with no chordal edge at Γ_{inn}. An ordering $\pi = \{v_1, \dots, v_n\}$ of the vertices of $G \backslash \Gamma_{inn}$ is called a* canonical ordering *if it satisfies:*

- *For each $k \in [0..n]$ the map G_k induced by Γ_{inn} and by the vertices $\{v_1, \dots, v_k\}$ is a cylindrical triangulation. The boundary outer face of G_k is denoted B_k.*
- *For each $k \in [1..n]$, the vertex v_k is on B_k, and its neighbours in G_{k-1} are consecutive on B_{k-1} (see Fig. 3.12).*

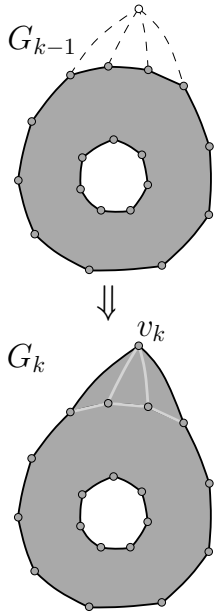


Figure 3.12: From G_{k-1} to G_k in a canonical ordering for a cylindrical simple triangulation (annular representation, only the boundaries and next added vertex and added edges are shown).

The notion of canonical ordering makes it possible to construct a cylindrical triangulation G incrementally, starting from $G_0 = \Gamma_{inn}$ and adding one vertex at each step. While in the planar case, one starts with G_0 being an edge, here one starts with G_0 being a cycle, seen as a cylindrical map with no internal face. The computation of such an ordering is done by a shelling procedure, similar to the one considered in Section 2.2.2. At each step the graph formed by the remaining vertices is a cylindrical triangulation, the inner boundary remains Γ_{inn} all the way, while the outer boundary B_k (initially Γ_{ext}) has its contour getting closer to Γ_{inn}. A vertex $v \in B_k$ is *free* if v is incident to no chord of B_k and if $v \notin \Gamma_{inn}$. The shelling procedure goes as follows, with n the number of vertices in $G \backslash \Gamma_{inn}$:

> for k from n to 1, choose a free vertex v on B_k, assign $v_k \leftarrow v$, and then delete v together with all its incident edges.

The existence of free vertices follows from the same argument as in the planar case [Breoo]. First, since there is no chord at Γ_{inn}, then as long as $B_k \neq \Gamma_{inn}$ there is at least one vertex on $B_k \backslash \Gamma_{inn}$. If there is no chord for B_k, then any vertex $v \in B_k \backslash \Gamma_{inn}$ is free. If there is at least one chord $e = \{u, v\}$ for B_k, let P_e be the path connecting u and v on B_k such that the cycle $P_e + e$ does not enclose the inner boundary-face in the annular representation, and let d_e be the length of P_e (note that $d_e \geqslant 2$). Let $e = \{u, v\}$ be a chord such that d_e is smallest possible. Then any vertex in $P_e \backslash \{u, v\}$ is free. Since there exists a free vertex at each step, the procedure terminates as stated below:

The linear time complexity of the shelling procedure is justified in [CDF18b]. This algorithm is rather simple to implement and fast in practice: as by-product this leads to efficiently compute toroidal Schnyder woods as illustrated in Section 3.4.3.

Proposition 3.7 (Castelli Aleardi, Devillers, and Fusy [CDF18b]) *Any cylindrical simple triangulation G with no chordal edge at Γ_{inn} admits a canonical ordering that can be computed in linear time by a shelling procedure.*

UNDERLYING FOREST AND DUAL FOREST. Given a cylindrical simple triangulation G with no chordal edge at Γ_{inn} endowed with a canonical ordering π, we define the *underlying forest* F for π as the oriented subgraph of G where each vertex $v \in \Gamma_{ext}$ has outdegree 0, and where each $v \notin \Gamma_{ext}$ has exactly one outgoing edge, which is connected to the adjacent vertex u of v of largest label in π. The forest F can be computed on the fly during the

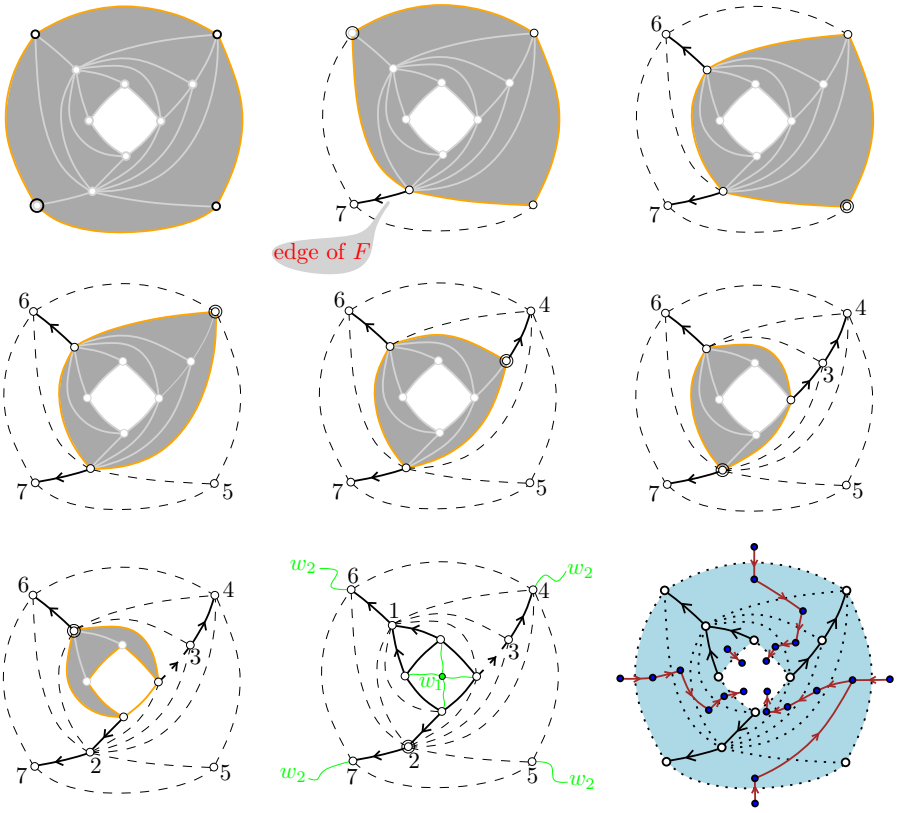


Figure 3.13: Shelling procedure to compute a canonical ordering of a cylindrical simple triangulation (at each step the next shelled vertex is surrounded). The underlying forest is computed on the fly; the last drawing shows the underlying forest superimposed with the dual forest.

shelling procedure: when processing an admissible vertex v_k, for each neighbour v of v_k such that $v \notin B_k$, add the edge $\{v, v_k\}$ to F, and orient it from v to v_k. Since the edges are oriented in increasing labels, F is an oriented forest; it spans all vertices of G and has its roots on Γ_{ext}. The *augmented map* $\widehat{G}$ ($\widehat{G}$ has to be seen as a map on the sphere) is obtained from G by adding a vertex w_1 inside B_{inn}, a vertex w_2 inside B_{ext}, and connecting all vertices around B_{inn} to w_1 and all vertices around B_{ext} to w_2, thus triangulating the interiors of B_{inn} and B_{ext}. We denote by $\widehat{F}$ the forest F plus all edges incident to w_1 and all edges incident to w_2. The *dual forest* F^* for π is defined as the graph formed by the vertices of $\widehat{G}^*$ (the dual of $\widehat{G}$) and by the edges of $\widehat{G}^*$ that are dual to edges not in $\widehat{F}$. Since $\widehat{F}$ is a spanning connected subgraph of $\widehat{G}$, F^* is a spanning forest of $\widehat{G}^*$. Precisely, each of the trees (connected components) of F^* is rooted at a vertex "in front of" each edge of Γ_{inn}, and the edges of the tree can be oriented toward this root-vertex, see Fig. 3.13 bottom right. Each edge e^* of F^* is in a certain tree-component T^* rooted at a vertex V_0 in front of a certain edge of Γ_{inn}. Let P be the path from e^* to V_0 in T^*; P is shortly called *the root-path of* e^*.

FROM CANONICAL ORDERINGS TO EDGE COLORING AND ORIENTATIONS. Once we have a notion of canonical ordering for cylindrical triangulations we can easily get a partition of the edges (except the ones on Γ_{inn}) into three sets of oriented and colored edges. We proceed in the usual way, already

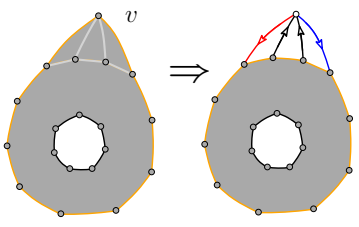


Figure 3.14: Coloring rule associated to the removal of a vertex v.

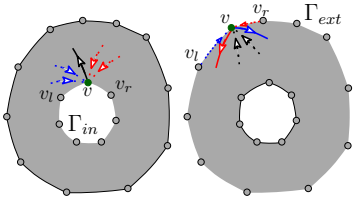


Figure 3.15: Local conditions for vertices on Γ_{inn} (left) and vertices on Γ_{ext} (right).

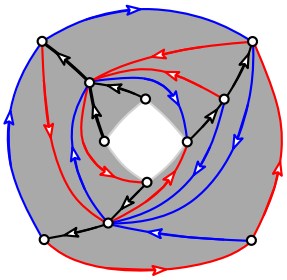


Figure 3.16: A cylindrical triangulation endowed with an edge coloring/orientation according to Corollary 3.8.

described for the planar and genus g case, by defining a coloring rule that assign colors and orientations to edges during the shelling process described above. This is better described by the statement below which is a by-product of Proposition 3.7:

Corollary 3.8 *Let* G *be a cylindrical simple triangulation with no chordal edge at* Γ_{inn}. *Then it is possible to compute in linear time an orientation and coloring of all edges in* $G \setminus \Gamma_{inn}$ *satisfying the following conditions:*

- ***local condition for inner vertices:*** *each inner vertex* v *has exactly three outgoing edges (one for each color) and the edges incident to* v *satisfy the local Schnyder condition as in the planar case;*
- ***local condition for*** Γ_{ext} *(see right picture in Fig. 3.15): each vertex* $v \in \Gamma_{ext}$ *has exactly two outgoing edges (of color* 0 *and* 1*); moreover, all the edges between the right and left neighbors of* v *on* Γ_{ext}, *are listed starting from* (v, v_r) *to* (v, v_l) *in cw direction around* v *as follows:*

$$\mathrm{Seq}(\mathrm{In}\,0), \mathrm{Out}\,1, \mathrm{Seq}(\mathrm{In}\,2), \mathrm{Out}\,0, \mathrm{Seq}(\mathrm{In}\,1).$$

- ***local condition for*** Γ_{inn} *(see left picture in Fig. 3.15): each vertex* $v \in \Gamma_{inn}$ *has exactly one outgoing edge (of color* 2*), and all incident edges between* (v, v_l) *and* (v, v_r) *are listed in cw direction as follows:*

$$\mathrm{Seq}(\mathrm{In}\,1), \mathrm{Out}\,2, \mathrm{Seq}(\mathrm{In}\,0).$$

Observe that the edges of color 2 define the underlying forest F described above. Similarly, the edges of color 0 and 1 define two forests spanning all vertices of G, whose connected components are ending at the vertices of Γ_{inn} (as illustrated in the example of Fig. 3.16).

3.4 SCHNYDER WOODS FOR TOROIDAL TRIANGULATIONS

The first generalizations [BGL05; CAFL09] of Schnyder woods were motivated by applications involving graph encoding problems, aiming to preserve their global spanning properties. Since then the study of Schnyder woods for non planar graphs have a attracted a lot of attention. This is particularly true in the toroidal case, for which some elegant generalizations led to several applications, ranging from graph drawing to bijective encodings.

3.4.1 *The definition of Gonçalves and Lévêque [GL14] (for the triangulated case)*

The key idea underlying the definition proposed by Gonçalves and Lévêque [GL14] is that Euler formula says that the number of edges is exactly $m = 3n$ on the torus. This means that in principle it could be possible to endow a triangulation with an edge orientation where, hopefully, every vertex satisfies the local Schnyder rule. So, in the toroidal case, the hope is to get rid of the special role of the three outer vertices lying on the root face, and to keep the symmetry of the underlying edge partition (observe that this symmetry is lost in the definition of g-Schnyder woods given in Section 3.2.1).

Finally, it turns out that a combinatorial structure satisfying the requirements above exists in the toroidal case, leading to the following definition [1] illustrated in Fig. 3.17 :

1 Observe that the definition in [GL14] is more general and extends the notion of Schnyder woods for the 3-connected case.

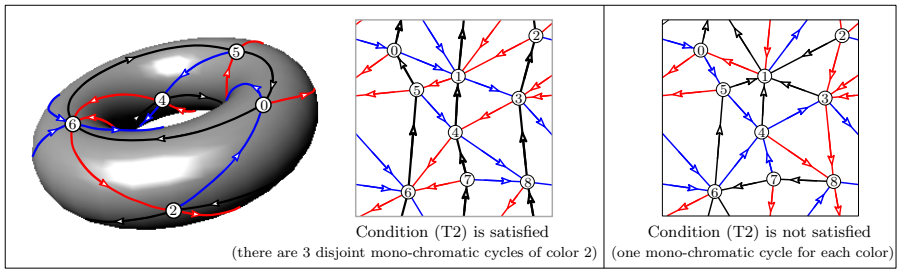


Figure 3.17: Toroidal Schnyder woods for simple triangulations satisfying the local Schnyder condition (T1) of Definition 3.9 around every vertex. (left) the edge orientation and coloring satisfies the crossing condition (T2) (but there are several disjoint mono-chromatic cycles). (right) Another edge coloring and orientation of the same triangulation: each color defines one mono-chromatic cycle, but the crossing condition (T2) is not satisfied.

Definition 3.9 (Gonçalves and Lévêque [GL14]) *Let $\mathcal{T}$ be a toroidal triangulation. A* Schnyder wood *of $\mathcal{T}$ is an orientation and labeling, with labels in $\{0,1,2\}$ of the edges of $\mathcal{T}$ so as to satisfy the following condition:*

- ***(T1) local Schnyder condition (for all vertices):*** *For each vertex v, the edges incident to v in counterclockwise (*ccw*) order are: one outgoing edge colored 2, zero or more incoming edges colored 1, one outgoing edge colored 0, zero or more incoming edges colored 2, one outgoing edge colored 1, and zero or more incoming edges colored 0;*

The Schnyder wood is said to be crossing *if it satisfies the further condition below*

- ***(T2) Crossing condition:*** *Every monochromatic cycle of color i intersects at least one monochromatic cycle of color $i-1$ and at least one monochromatic cycle of color $i+1$.*

Here we adopt the formulation for the triangulated case given in [DGL17]: observe that the crossing condition is included in the definition of toroidal Schnyder woods originally proposed in [GL14]

The definition above is particularly elegant since it preserves the symmetry of the three colors, so there is a natural correspondence between a toroidal Schnyder wood and the underlying 3-orientation (up to a cyclic permutation of the colors). In principle the global spanning property of Schnyder wood is lost, since the edges of a same color could define a non connected graph: as illustrated by left example in Fig. 3.17, the graph induced by one color could consist of several disjoint connected components, but it is possible to prove that each component contain exactly one directed cycle. More precisely, Gonçalves and Lévêque (see Theorem 7 in [GL14]) show that in a toroidal crossing Schnyder wood all mono-chromatic cycles of the same color are non-contractible and homotopic, and they do not intersect each other; so condition (T2) becomes even stronger, since one can show that every pair of mono-chromatic cycles Γ_i and Γ_j (for $i \neq j$) are not homotopic and must intersect.

The condition (T2) is particularly interesting, leading to several applications. For instance, this condition is crucial for obtaining a periodic grid drawing of toroidal triangulations that extends the region counting approach of planar Schnyder drawings [GL14]. Moreover, if one consider, for a given triangulation, the set of 3-orientations that correspond to crossing Schnyder woods then it is possible to define a distributive lattice structure similarly to the planar case (we refer to [KGL19] for more details).

3.4.2 *Existence of toroidal Schnyder woods*

As shown in [GL14] it is possible to show the existence for the general case of essentially 3-connected toroidal maps making use of the approach based on edge contractions. The main idea is to perform a sequence of contractions until the resulting graph has few vertices: the Schnyder wood can be recovered by assigning colors after the edge expansions (in reverse order). The validity of this approach relies on very involved case analysis (observe that there are no special outer vertices in the definition of toroidal Schnyder woods): unlike planar Schnyder woods, in the toroidal case one has to distinguish many cases of edge contractions in order to preserve the structure of monochromatic cycles, which is a non local property. For the triangulated (simple) case there is another simpler proof, as sketched below.

As far as we know there are no practical implementations of the procedure based on edge contractions for computing toroidal Schnyder woods, even for triangulations, for which the case analysis is much simpler. Observe that obtaining a linear-time implementation requires to choose contractible edges in a fast way (which is possible, but not trivial to implement).

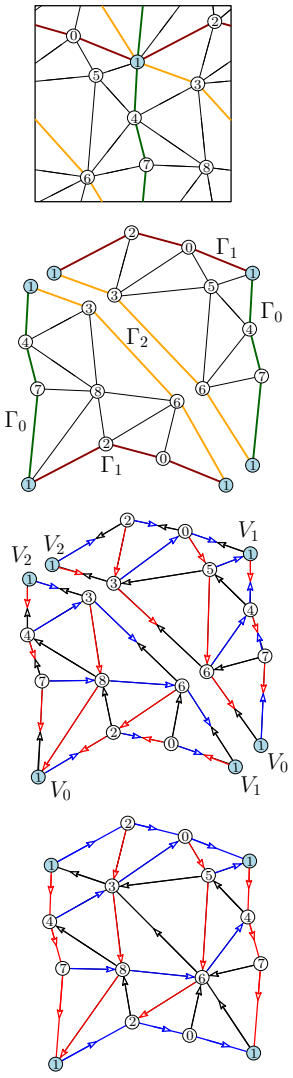


Figure 3.18: Existence of toroidal Schnyder woods for simple toroidal triangulations [GL14].

TOROIDAL SCHNYDER WOODS FOR SIMPLE TRIANGULATIONS. For the case of simple triangulations (no loops and no multiple edges) Gonçalves and Lévêque [GL14] propose another proof of existence of toroidal Schnyder woods: it suffices to apply a planarization approach relying on Theorem 2.2.

The idea, illustrated in Fig. 3.18, consists to cut the input triangulation $\mathcal{T}$ along three non-contractible (and non-homotopic) cycles Γ_0, Γ_1 and Γ_2 sharing a common vertex v. As result one obtains a partition of the faces of $\mathcal{T}$ into two quasi-triangulations G_1 and G_2 which are internally 3-connected and have 3 copies of vertex v, denoted V_0, V_1 and V_2, on their boundaries. It is then possible to endow both G_1 and G_2 with a planar Schnyder wood, where all inner vertices satisfy the local Schnyder condition and the boundary edges have two opposite orientations (for more details please refer to [Felo1; Felo3]). Now it suffices to assign colors and orientations to the edges according to the following rule: assign color i to a boundary edge appearing on the path Γ_i (oriented toward V_i) for $i \in \{0, 1, 2\}$. As observed by Gonçalves and Lévêque [GL14], this ensures that after gluing together G_1 and G_2 the resulting edge coloring and orientation corresponds to a valid toroidal crossing Schnyder wood as stated below:

Proposition 3.10 (Gonçalves and Lévêque [GL14]) *Any simple toroidal triangulation $\mathcal{T}$ admits a toroidal crossing Schnyder wood. Moreover, such a Schnyder wood contains three monochromatic cycles (one for each color) all intersecting at one common vertex, while being pairwise disjoint otherwise.*

Observe that the statement above does not guarantee that the edges of the same color form a connected graph: in principle the corresponding Schnyder wood could contain several parallel monochromatic cycles of the same color.

Unfortunately, as pointed out in [Lév16], it is not clear how to turn the procedure above into a linear-time algorithm (the preliminary step requires Theorem 2.2 which relies on a result involving disjoint paths by Robertson and Seymour [RS86]).

Open questions

As far as we know there are a few interesting questions concerning toroidal and higher genus Schnyder woods left open by [GL14; AGK16; KGL19] (see also [Lév16] for more details).

> *"Does a simple toroidal triangulation admit a Schnyder wood such that there is just one monochromatic cycle per color?"*

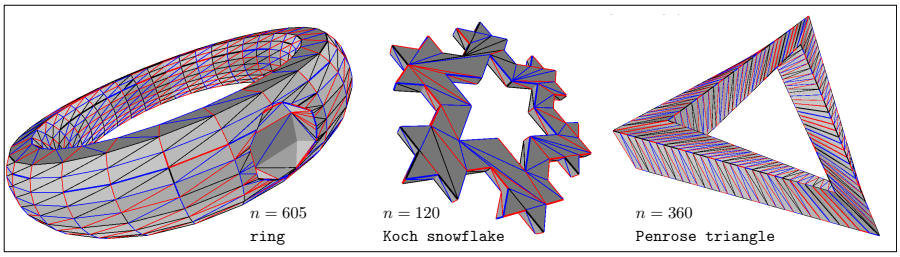


Figure 3.19: A few triangles meshes of genus 1 endowed with toroidal Schnyder woods satisfying condition (T1) of Definition 3.9. They have been obtained with our implementation of the shelling procedure described by Corollary 3.11.

> *"Moreover, can one require that monochromatic cycles all intersect on one vertex and they are pairwise disjoint otherwise?"*
> Is it possible *"to generalize the shelling order method for the torus"*?

A further interesting open question involves the existence of generalized Schnyder woods for maps of arbitrary genus. A partial answer to this question has been very recently provided by Jason Suagee [Sua21] in its PhD dissertation: unfortunately the assumptions are quite strong, as the existence of generalized Schnyder woods is guaranteed only for triangulations having very large edge-width (refer to Section 6.1 for a few more details).

We will address some of the problems above, providing theoretical (Section 3.4.3) and experimental (Section 6.1) partial answers to these questions.

3.4.3 *Our contribution: toroidal Schnyder woods via vertex shellings*

It is interesting to observe that our incremental algorithm for computing canonical orderings of cylindrical triangulations can be turned into an efficient and simple procedure for computing toroidal Schnyder woods satisfying the condition (T1) of Definition 3.9.

COMPUTING A NON-CONTRACTIBLE CYCLE. We first compute a directed non-contractible cycle Γ of the input (simple) toroidal triangulation $\mathcal{T}$ such as there are no chordal edges on the right side of Γ (observe that we do not care about the length of this cycle). For instance, such a cycle can be derived by our shelling procedure described in Section 3.2.2. Just perform a sequence of `conquer` operations until a *split edge* $e = (u, v)$ is found. Then we can simply consider the two paths $\Pi_1(u) = \{u, \dots, a, \dots, V_2\}$ and $\Pi_2(v) = \{v, \dots, a, \dots, V_2\}$, where a is the common ancestor of u and v in the tree T_2: because of the Schnyder local condition both paths are chord-free.

Assume that v is preceding u on the boundary of the conquered region (which is planar), i.e., v is closer to vertex V_0. The cycle Γ is obtained concatenating the subpath $\Pi'_1(u) = \{u, \dots, a\}$ with the subpath $\Pi'^{-1}_2(v) = \{a, \dots, v\}$ (where we reverse the direction of the edges of $\Pi_2(v)$) and finally adding the edge (v, u). As illustrated in Fig. 3.20, at the exception of u and v all vertices of Γ are included in a planar region C (whose boundary is a simple contractible cycle): the local Schnyder planar condition ensures that there are no chords on the right side of Γ inside C. The only two vertices of Γ appearing on the boundary of C are u and v: a chordal edge between them would yield to the existence of a multiple edge (a contradiction, as by assumption the input triangulation is simple). Observe that the cycle Γ we

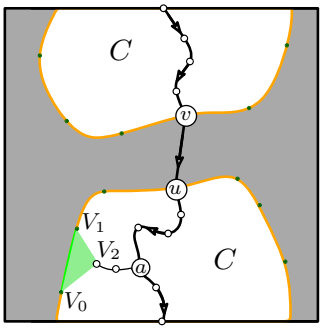


Figure 3.20: Computation of a non-contractible (oriented) cycle Γ with no chordal edges on its right side.

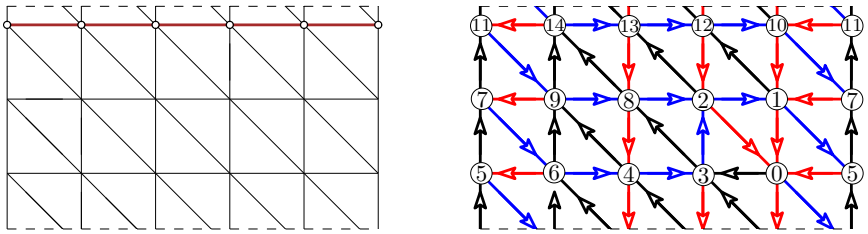


Figure 3.22: Computation of toroidal Schnyder woods via vertex shellings: the input toroidal triangulation is cut along a non contractible cycle Γ with no chordal edges on one side (left picture), which allows us to compute a cylindrical canonical ordering (vertex labels reflect the shelling order). The resulting edge coloring (right picture) does not satisfy the crossing condition (T2): for instance there are two parallel cycles of color 0 and 2 that are not intersecting.

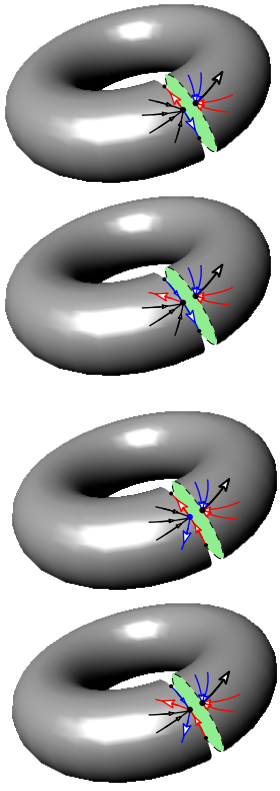


Figure 3.21: Local condition for vertices on the non-contractible cycle Γ: every vertex has an outgoing edge of color 2 on the right side of Γ; the outgoing edges of color 0 and 1 are on the left side, possibly lying on Γ.

obtained in this way is non-contractible since it is included in the cut-graph G_2 of $\mathcal{T}$ defined in Section 3.2.3.

FROM CYLINDRICAL CANONICAL ORDERING TO TOROIDAL SCHNYDER WOODS. Given a non-contractible cycle Γ (with no chords on its right side) as described above we can planarize $\mathcal{T}$ cutting along Γ obtaining a cylindrical triangulation G whose inner inner Γ_{inn} is chord-free: we can thus running our algorithm for cylindrical canonical orderings and endow G with an edge coloring satisfying the constraints of Corollary 3.8. Let us consider the edge coloring and orientation of $\mathcal{T}$ obtained by gluing Γ_{inn} and Γ_{ext}, where the edges of Γ are naturally inheriting the edge coloring and orientation from Γ_{ext}. It is easy to see the we obtain a 3-orientation, as a simple by-product of Corollary 3.8. Moreover for every vertex $v \in \Gamma$ one can check that the edge coloring satisfies condition (T1) by distinguishing a few cases, depending on the colorations of edges incident to v: as illustrated in Fig. 3.21 after gluing the two boundary cycles Γ_{inn} and Γ_{ext} the local Schnyder condition is satisfied (all other vertices, not lying on Γ, satisfy the same condition as in the planar case).

Corollary 3.11 *Let $\mathcal{T}$ be a simple toroidal triangulation of size n. Then it is possible to compute in linear-time a Schnyder wood of $\mathcal{T}$ satisfying the condition (T1) for all vertices.*

Unfortunately we have to point out that the edge coloring provided by Corollary 3.11 is not guaranteed to satisfy the condition (T2). There are examples computed by our algorithm such that the resulting Schnyder wood contains monochromatic cycles of different colors that are no intersecting: in the example of Fig. 3.22 there is a black cycle not intersecting any monochromatic red (it is actually parallel to another red cycle).

As one would expect the approach sketched above leads to a simple and very fast implementation: the pictures in Fig. 3.19 show a few triangle meshes of genus 1 endowed with a toroidal Schnyder wood. Let us mention that, to our knowledge, there are no existing implementations of the algorithms for computing toroidal Schnyder woods and higher genus 3-orientations described in [GL14; DGL17; AGK16; KGL19; Sua21].

4 GRAPH ENCODING AND COMPACT DATA STRUCTURES

The problem of encoding and representing graphs involves a broad spectrum of technics and applications, and has been addressed in the last four decades in several distinct domains including computational geometry, geometry processing and computer graphics, enumerative combinatorics, graph algorithms, data compression and complex networks.

The problem of encoding the combinatorics of graphs embedded on surfaces is part of the so-called *mesh compression problem*: we refer to the surveys by Alliez and Gotsman [AG05] and by Maglo et al. [Mag+15] that provide a comprehensive overview of the results developed in the geometry processing community. For the reader who is interested in the design of compact data structures and compression schemes dealing with more general classes of graphs (such as *complex networks*) we suggest the recent survey by Besta and Hoefler [BH18].

In this chapter we consider the problem of encoding and compactly representing graphs which are (locally) planar and we focus on works that make use of *tree-based* decompositions. In particular, the use of Schnyder woods and related combinatorial structures has led to the most efficient compression schemes and compact data structures provided with theoretical guarantees which are known so far.

The contributions of this chapter [1] are originated from the collaborations with O. Devillers, É. Fusy and T. Lewiner [CAFL09; CFL10; CD18].

4.1 RELATED WORKS

4.1.1 *Tree-based encodings*

The use of spanning trees decompositions dates back to the pioneristic work of Turan [Tur84], who proposed a general way for efficiently representing the combinatorics of a planar graph with a linear number of bits. The idea is very simple (and illustrated on a small example in Fig. 4.1): given a planar graph G with n vertices one has just to chose any (arbitrary) spanning tree T of G and to encode the edges appearing both in T and $G \setminus T$. Because of the planarity of G and of its dual, T and $G \setminus T$ can be both represented with two balanced parenthesis words whose size are roughly $2n$. Unfortunately, in order to recover from these two words the original graph G, the two encodings of T and $G \setminus T$ must be interleaved. Thus we need a multiple parenthesis word (consisting of two types of open/closed parentheses) whose length is $2e$: this yields an encoding on $\approx 4e$ bits, since each edge is encoded by a pair of open/closed parenthesis (or brackets), and we need two bits to encode each of the 4 symbols. Planar triangulations can be thus encoded with at most $12n$ bits.

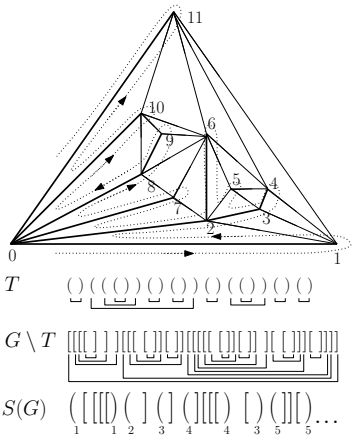


Figure 4.1: To encode a planar graph G with n vertices the Turan encoding [Tur84] makes use of an arbitrary spanning tree T of G: both T and $G \setminus T$ can be represented with balanced parenthesis words of size at most $2n$, obtained performing a ccw traversal of the contour of T (rooted at vertex 0).

The framework proposed by Turan is quite general and thus not really efficient, at least when compared to the current state-of-the-art. Quite interestingly, as pointed out by Isenburg [IS05], many compression schemes de-

1 This chapter does not include my works [CDS05a; CDS05b; CDS08; CDM11; Bar+12] on the so-called *succinct representations* (sometimes referred to as *succinct encodings* or *succinct data structures*), most of which have been developed during my PhD.

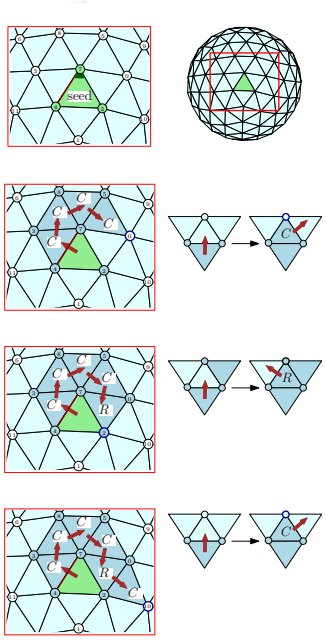


Figure 4.2: A few steps performed by *Edgebreaker* [Ros99]. The traversal starts from an initial *seed* (green) face and performs a depth-first visit of the dual graph (blue triangles belong to the visited region).

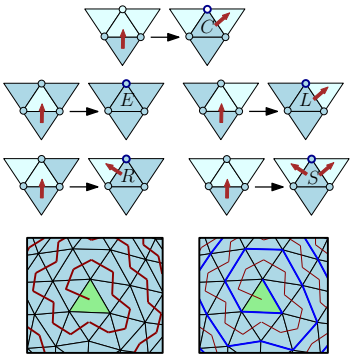


Figure 4.3: *Edgebreaker* encoding sequence: there are five cases to consider, depending on the adjacencies of each traversed (blue) triangle. These cases are described by five symbols $\{C, L, E, R, S\}$: the most frequent case, encoded by the symbol C, is when the visited triangle is incident to a new unvisited vertex, which occurs $n-3$ times. If we use 1 bit for symbol C and 3 bits for symbols $\{L, E, R, S\}$, the resulting encoding sequence (whose length is $m-1 < 2n$) can be compressed using at most $1 \cdot (n-3) + 3 \cdot n \approx 4n$ bits. Observe that in the planar case a more sophisticated encoding of the *CLERS* sequence leads to a better compression rate of 3.67 bpv [KR99].

veloped since the early nineties in both the geometry processing and graph algorithms communities fits into the framework above. For instance, the popular *Edgebreaker* compression scheme proposed by Rossignac [Ros99] relies on an incremental *growing-region* approach (illustrated in Fig. 4.2). The algorithm performs an incremental traversal of the dual graph starting from an initial *seed* face: Edgebreaker makes use only of five symbols in order to encode a spanning tree of the dual graph and its complement, a spanning tree of the primal graph (refer to Fig. 4.3): a triangulation with n vertices can be encoded using at most $4n$ bits.

4.1.2 *Optimal encodings achieving information-theory lower bounds*

PLANAR CASE. The *Edgebreaker* encoding above is both simple and quite compact, since its compression rate of 4 bpv (*bits per vertex*) is not far from the information-theory asymptotic lower bound given by

$$\frac{1}{n}\log_2|\mathcal{T}_n| = \frac{1}{n}\frac{2(4n-3)!}{(3n-1)!(n)!} \approx \log_2\left(4^4/3^3\right) \approx 3.2451 \text{ bpv} \qquad (4.1)$$

(where $|\mathcal{T}_n|$ is the number of rooted planar 3-connected triangulations with $n+2$ vertices, found by Tutte [Tut62]).

The bound above is attained by the nice bijective construction due to Poulalhon and Schaeffer [PS06], which relies on the correspondence between minimal Schnyder woods and a special class of spanning trees (having two stems per node); this bijection provides a combinatorial proof of the counting formula for $|\mathcal{T}_n|$, also yielding efficient procedures for uniform random sampling (this construction will be sketched in Section 4.3).

HIGHER GENUS CASE. In the higher genus case there does not exist an exact enumeration formula, nevertheless an asymptotic estimate [Gao93] of the number of genus g rooted triangulations with n vertices leads to the information theory lower bound of $3.245n + \Omega(g \log n)$, i.e., the exponential growth rate is the same in every genus.

For the genus 1 case, a special class of toroidal Schnyder woods allowed Despré, Gonçalves, and Lévêque [DGL17] to generalize the Poulalhon and Schaeffer bijection, in order to obtain an optimal encoder for toroidal triangulations. Unfortunately, in the higher genus g case there are still several questions that remain open involving the existence of Schnyder woods: bijective constructions based on special spanning trees are still to be found.

4.2 ENCODING TRIANGULATIONS IN ARBITRARY GENUS

A main application that motivated our definition of g-Schnyder woods was to extend to higher genus the encoding procedure of [HKL99; BB07] based on Schnyder woods: while not being asymptotically optimal, this approach is quite simple and yields a linear-time encoding achieving the same bound as *Edgebreaker*.

The encoding based on Schnyder woods turns out to be extremely useful for dealing in practice with compact data structures: as we will illustrate in Section 4.4. A simple adaptation leads to a very efficient way of decoding a compact data structure from compressed format in a streamable way: this leads to drastically reduce the memory requirements consumed in the preprocessing constructive phase.

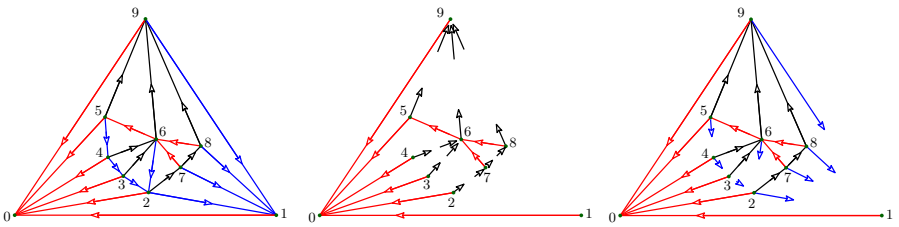


Figure 4.4: Encoding of a plane triangulation $\mathcal{T}$ (endowed with a Schnyder wood orientation): $\mathcal{T}$ is encoded by a pair of binary words, encoding respectively the red tree $\overline{T}_0$ and the tails of black edges (center). The information concerning blue edges is redundant and can fully retrieved by applying the local Schnyder condition (right).

THE PLANAR CASE. In the planar case, Schnyder woods yield a simple encoding procedure for triangulations, as described in [HKL99] and more recently in [BB07]. For a triangulation with n vertices, W, W' have length at most $2n$, hence the coding word has total length at most $4n$.

More precisely, a planar Schnyder wood with $n$vertices is encoded by two parenthesis words W, W' of respective lengths $2n-2$ and $2n-6$. Let $\overline{T_0}$ be the tree T_0 plus the two outer edges incident to V_0. Call θ the corner incident to V_0 in the outer face. The first word W is the parenthesis word (also called Dyck word) that encodes the tree $\overline{T_0}$, that is, W is obtained from a ccw walk (i.e., the walker has the infinite face on its right) around $\overline{T_0}$ starting at θ, writing an opening parenthesis at the first traversal of an edge of $\overline{T_0}$ (away from the root) and a closing parenthesis at the second traversal (toward the root). The second word W' is obtained from the same walk around T_0, but W' encodes the edges that are not in $\overline{T_0}$, i.e., the edges of color 1 and 2. Precisely, during the traversal, write a '0' symbol in W' each time an outgoing edge in color 1 is crossed and write a '1' symbol in W' each time an ingoing edge of color 2 is crossed (this corresponds to a unary encoding of the ingoing degree of edges of color 2 at each vertex $v \in V \setminus \{V_0, V_1\}$).

The string below corresponds to the encoding of the triangulation in Fig. 4.4 (vertices are ordered according to the ccw-DFS traversal of $\overline{T}_0$):

( ) ( ) ( ) ( ) ((( ) ( ))) ( ) 00001101010111

For the sake of clarity, we can decorate this code by vertex indices:

$(_1\ )_1\ (_2\ )_2\ (_3\ )_3\ (_4\ )_4\ (_5\ (_6\ (_7\ )_7\ (_8\ )_8\ )_6\ )_5\ (_9\ )_9\ 0_2\ 0_3\ 0_4\ 0_5\ 110_6\ 10_7\ 10_8\ 111$

In the example above the black word encodes the in-black-degrees of vertices, which are respectively 0, 0, 0, 0, 0, 0, 2, 1, 1, 3.

While not being asymptotical optimal, this encoding has the advantage of being extremely simple to implement and fast in practice: as confirmed by the experiments reported in Fig. 4.5 our Java implementation is able to process between 2.2M and 3.7M vertices per second (which makes this approach highly competitive with prior works on mesh compression).

Remark. Observe that, in the planar case, because of the symmetry of the definition of Schnyder woods in the three colors, one could perform a contour traversal of the spanning tree T_1 (or T_2) instead of T_0, and the traversal may be in either cw or ccw direction (similar arguments would apply).

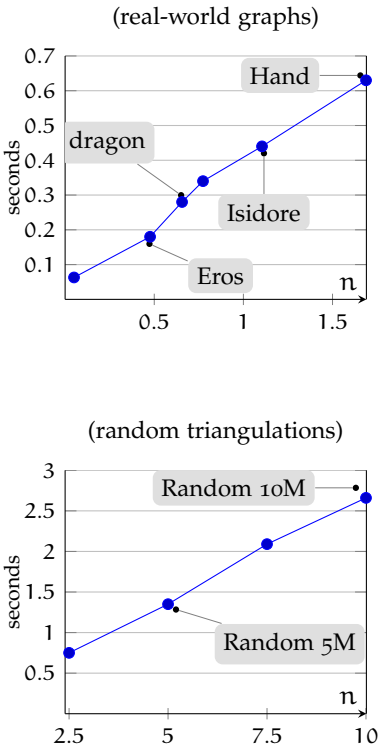


Figure 4.5: Runtime cost of the algorithm for encoding planar triangulations (it includes the pre-processing time, for computing the Schnyder wood). Timings are expressed in seconds as a function of the size (millions of vertices). In our tests we allocate 6GB of RAM memory for running our Java implementation [CD18].

4.2.1 *Encoding in higher genus*

To encode a genus g triangulation $\mathcal{T}$ of size n we proceed in a similar way as in the planar case except that we endow $\mathcal{T}$ with a g-Schnyder wood com-

puted in $O(n)$ time according to Theorem 3.4, and thus we have to deal with the special edges involved in Definition 3.1.

Let $\overline{T_2}$ be the spanning tree of $\mathcal{T}$ consisting of the edges in color 2 plus the two edges $\{V_0, V_2\}$ and $\{V_1, V_2\}$ (refer to Fig. 4.6). Let G_2 be the graph $\overline{T_2}$ plus the $2g$ special edges (according to Proposition 3.5 this is a cut-graph of $\mathcal{T}$). As in previous section we classically encode G_2 as the Dyck word W for $\overline{T_2}$, augmented by $2g$ memory blocks, each of size $O(\log(n))$ bits, so as to locate the two extremities of each special edge: in each memory block we also store the colors and directions of the two sides of the special edge (as depicted in the middle picture of Fig. 4.6). Hence G_2 is encoded by a word W of length $2n - 2 + O(g \log(n))$. The encoding of the g-Schnyder wood is completed by a second binary word W' that is obtained from a clockwise walk along the (unique) face of G_2 (cw means that the face is on the right of the walker) starting at the corner θ incident to V_2 in the root-face. Along this walk, we write a $'0'$ bit when crossing a non-special outgoing edge of color 0 and we write a $'1'$ bit when crossing a non-special ingoing edge of color 1 (this corresponds to encode the ingoing degree of color 1 edges). Since there are $2n - 6 + 4g$ non-special edges of color 0 or 1, the word W' has length $2n - 6 + 4g$. Therefore the pair of words (W, W') is of total length $4n + O(g \log(n))$. Observe that these words can be obtained in time $O((n+g)g)$ from a g-Schnyder wood on $\mathcal{T}$, and the g-Schnyder wood itself can be computed in time $O((n+g)g)$.

The correctness of this procedure – the fact that the pair (W, W') actually encodes the g-Schnyder wood (and in particular the triangulation) – relies on the two lemmas below (the proofs are omitted here and can be found in [CAFL09]). First Lemma states that the information given by non-special edges of color 0 is redundant.

Lemma 4.1 *Let $\mathcal{T}$ be a triangulation endowed with a g-Schnyder wood. Then the g-Schnyder wood can be recovered after the deletion process that consists in removing all the non-special edges of color 0.*

Lemma 4.2 *Let $\mathcal{T}$ be a triangulation endowed with a g-Schnyder wood and consider the corresponding cut-graph G_2. Let θ be the corner incident to V_2 in the root-face (θ is also a corner of G_2). Let e be a non-special edge of color 1. Then, during a cw walk along G_2 (i.e., with the unique face of G_2 on the right of the walker) starting at θ, the outgoing brin of e is crossed before the ingoing brin of e.*

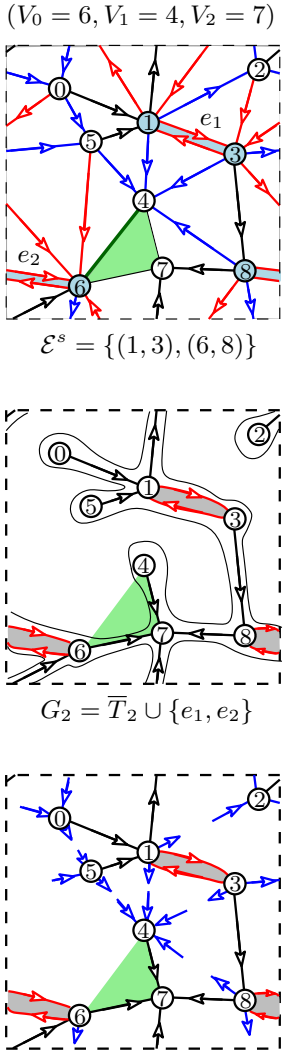


Figure 4.6: Encoding of a toroidal triangulation using our scheme based on g-Schnyder woods.

DECODING. We can now describe how to reconstruct the Schnyder wood from the two words (W, W'). First, construct the cut-graph G_2 using W. Note that the directions of edges and colors of the two sides of each special edge of G_2 are known from W. Hence, by the local conditions of Schnyder woods, we can already insert the outgoing brins of color 0 or 1 that are non-special (a non-special brin is a brin of a non-special edge). The non-special outgoing brins of color 0 are ordered as $b_1, b_2, \ldots, b_k$ according to the order in which they are crossed during a cw walk along G_2 (i.e., with the unique face of G_2 on the right of the walker). Next, the word W' indicates where to insert the non-special ingoing brins of color 1. Precisely, factor W' as

$$W' = 1^{r_1} 0 1^{r_2} 0 1^{r_3} \ldots 0 1^{r_{k+1}},$$

where the integers r_i's are allowed to be zero. Then, for each $i \in [1..k]$, insert r_i ingoing brins of color 1 in the corner $(b_i, \text{follower}(b_i))$ (where the follower of a brin b is the next brin after b in cw order around its origin). And insert

r_{k+1} ingoing brins of color 1 in the corner incident to V_1 delimited to the right by $\{V_1, V_0\}$.

Afterwards, we use Lemma 4.2 to form the non-special edges of color 1. Write a parenthesis word π obtained from a cw walk along G_2 starting at θ, writing an opening parenthesis each time a non-special outgoing brin of color 1 is crossed and writing a closing parenthesis each time a nonspecial ingoing brin of color 1 is crossed. Then, Lemma 4.2 ensures that the matchings of π correspond to the non-special edges of color 1 in the Schnyder wood, so we just have to form the non-special edges of color 1 according to the matchings of π.

Finally, since the edges of color 0 are redundant (by Lemma 4.1), there is no ambiguity to insert the edges of color 0 at the end (i.e., complete the already inserted outgoing half-edges of color 0 into edges).

To conclude, the non-special edges of color 0 are redundant, the cut-graph can be encoded by a parenthesis word W of length $2n-2$ (for the tree $\overline{T_2}$) plus $O(g\log(n))$ bits of memory for the special edges, and the edges of color 1 can be inserted from a word W' of length $2n-6+4g$. Clearly the reconstruction of the Schnyder wood from (W, W') takes time $O((n+g)g)$, since it just consists in building the cut-graph G_2 and walking cw along G_2. All in all, we obtain the following result:

Theorem 4.3 (Castelli Aleardi, Fusy, and Lewiner [CAFL09]) *A triangulation of genus g with n vertices can be encoded—via a g-Schnyder wood—by a binary word of length at most $4n + O(g\log(n))$. Coding and decoding can be done in time $O((n+g)g)$.*

CONCLUDING REMARKS. While not asymptotically optimal, the encoding above matches the same compression rate of the *Edgebreaker* encoder in the genus g case, being very close to the optimal information theory bound. In the next section we will explain how to encode triangulations on surfaces of arbitrary genus (possibly having boundaries) matching the asymptotic optimal bound (Theorem 4.5). It is worth nothing that the solution of Theorem 4.3 is conceptually simpler to implement and faster: the encoding/decoding is done in $O((n+g)g)$ time, while the solution offered by Theorem 4.5 requires more than linear time to be computed.

Another advantage is that the encoding of Theorem 4.3 can be combined with other algorithmic tools to design a more sophisticated code that supports adjacency queries in $O(1)$ time, as done in [Chu+98; Bar+12] [2], or to obtain compact practical data structures admitting fast implementations (as the ones described in Section 4.4).

4.3 OUR CONTRIBUTION: OPTIMAL ENCODING OF TRIANGULATIONS WITH FIXED TOPOLOGY

In this section we sketch our solution [3] for optimally encoding a triangulation of arbitrary topology (possible having handles and boundaries). More precisely, we consider of class of *triangulations with fixed topology*, which are simple triangulated maps (no loops, no multiple edges) of genus g possibly having $b \geqslant 0$ boundaries: all faces are triangles, except for b marked *boundary* faces (also called *boundaries*) which are assumed to be pairwise disjoint (and without self-intersections). Such maps are rooted when there is

2 The results obtained in [Bar+12], a collaboration with J. Barbay, M. He and I.Munro, pertain to the design of succinct representations and are not included in this document.

3 The contribution of this section originates from Castelli Aleardi, Fusy, and Lewiner [CFL10].

a distinguished marked edge (the root) incident to a non-boundary (root) face.

Our solution relies on the combination of previous combinatorial tools with a planarizing strategy: the main idea consists in recursively cutting the surface along non contractible cycles in order to reduce its genus. Thus the problem of encoding a triangulation with fixed topology (given by the number of its boundaries and its genus) reduces to the problem of encoding a planar triangulation with boundaries, which can be solved with our extension of the Poulalhon and Schaeffer [PS06] bijection.

It is worth noting that more recently an elegant generalization of the Poulalhon and Schaeffer bijection to the genus 1 case has been proposed by Despré, Gonçalves, and Lévêque [DGL17], providing an optimal encoding for toroidal triangulations. Compared to the planar case with no boundary, our construction is not bijective but the encoding is still asymptotically optimal in the information theory sense, and allows us to deal with triangulations of arbitrary topology (bounded number of boundaries and handles): an advantage is that our strategy is rather simple and does not involve the computation of special crossing toroidal Schnyder woods as in [DGL17].

ORIENTATIONS AND CANONICAL SPANNING TREES. An α-orientation of a planar map M is *minimal* if there is no counter-clockwise directed cycle, and is *accessible* if from every vertex one can reach the root-vertex by an oriented path. To such an orientation O is associated the so-called *canonical spanning tree* for O [Ber07], which is the unique spanning tree T satisfying:

1. the edges of T are oriented toward the root-vertex,
2. every edge $e \in M\backslash T$ has on its right the interior of the unique cycle of $e + T$.

The canonical spanning tree can be computed in linear time (according to the number of edges) by a traversal algorithm [PS06; Ber07]. The *fully decorated spanning tree* F for O is obtained by cutting each edge $e \in M\backslash T$ in its middle, leaving an *outgoing stem* (incident to the origin of e) and an *ingoing stem* (incident to the end of e), see Figure 4.7(c). Property 2 ensures that O can be recovered from F, since the edges of $M\backslash T$ correspond to the matchings of the cyclic parenthesis word formed by the stems in clockwise order around the unique face of F (outgoing stems being seen as opening parentheses and ingoing stems as closing parentheses).

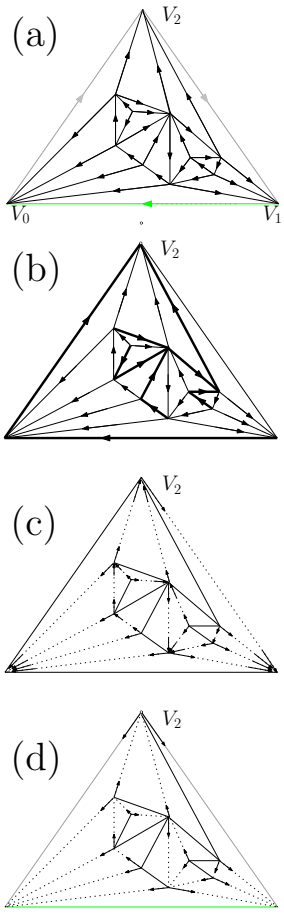


Figure 4.7: A planar rooted triangulation M endowed with its minimal 3-orientation (a), and the corresponding spanning tree T, rooted at V_2 (b).

4.3.1 *Encoding plane triangulations (Poulalhon and Schaeffer [PS06] bijection)*

Let $\mathcal{T}$ be a plane triangulation with $n+2$ vertices and root face $\{V_0, V_1, V_2\}$. The first step consists in computing a minimal Schnyder wood and the corresponding 3-orientation, denoted O, which is accessible with respect to every outer vertex. Then consider the corresponding fully decorated spanning tree, denoted F (see Fig. 4.7(c)). Since all faces are of degree 3, there is no loss of information in deleting ingoing stems (because there is a unique way to place the ingoing stems in such a way that the map obtained by matching outgoing with ingoing stems is a triangulation). One can also delete the branch (V_1, V_0), (V_0, V_2) without loss of information. The obtained tree with only outgoing stems is called the *reduced decorated spanning tree* R of M (see Fig. 4.7(d)): R is a rooted plane tree with 2 stems at each of the n nodes (the extremity of a stem being not considered as a node). As shown in [PS06],

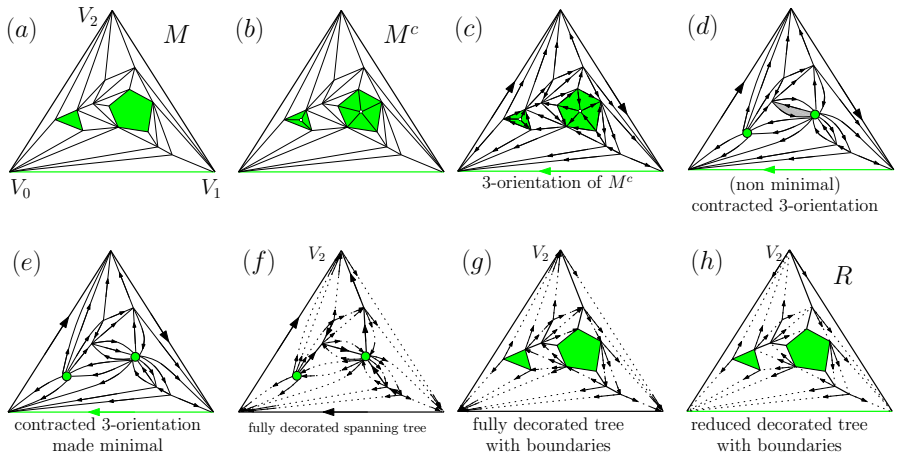


Figure 4.8: Correspondence between a triangulation M with 2 boundaries (a), and a decorated plane tree with 2 boundaries, spanning all vertices of $M \setminus \{V_0, V_1\}$ (h).

such a tree can be represented by a binary word of length $4n$ and weight n, which can be compressed leading to an encoding of size $\sim \log_2 \binom{4n}{n}$, matching the asymptotic bound given by (4.1).

4.3.2 *Encoding planar triangulations with multiple boundaries*

Here starts our contribution, which is to keep an optimal encoding scheme in case of boundaries. Let M be a plane triangulation with $b > 0$ boundaries, $n+2$ non boundary vertices, and k boundary vertices. Assume without loss of generality that an outer (non-boundary) face $\{V_0, V_1, V_2\}$ for M is fixed that does not touch any of the boundaries (if no such face exists, create it inside an arbitrary non-boundary face, this adds only 3 vertices and will have no effect on the length of the coding word asymptotically). Define the *completed triangulation* for M as the planar triangulation M^c obtained by adding a star in each boundary face (see Fig. 4.8(b)); M^c is a triangulation with $n+k+b$ vertices and no boundary. Endow M^c with Schnyder wood and consider the corresponding 3-orientation. Then contract each of the b stars $S_1, \ldots, S_b$ into a single so-called *special* vertex $s_1, \ldots, s_b$ (each contraction deletes the edges of the star and the edges on the contour of the corresponding boundary face). The orientation inherited by the 3-orientation of M^c is such that every non special vertex has outdegree 3, while every special vertex s_i has outdegree $k_i + 3$, with k_i the size of the corresponding boundary. The contracted orientation is still accessible with respect to vertices $\{V_0, V_1, V_2\}$, but it may not be minimal, even if coming from the minimal 3-orientation of M^c; indeed a path connecting two vertices on the same boundary might become a ccw circuit after contraction (as shown in Fig. 4.8(d)). However a procedure discussed in [BWoo] allows us to make —in linear time— the contracted orientation minimal (by successively reversing ccw circuits) while keeping the same outdegree at each vertex. Moreover the obtained minimal orientation is still accessible, because returning circuits does not affect accessibility. So we can now consider the fully decorated spanning tree for the orientation, see Fig. 4.8(f). Then we uncontract each special vertex back into the original boundary face and obtain a fully decorated plane tree with boundaries. Without loss of information we can delete ingoing stems and the branch (V_0, V_1), (V_0, V_2), to obtain the

so-called *reduced decorated plane tree with boundaries* R for M. The tree R belongs to the family $\mathcal{P}^{(b)}_{n,k}$ of plane trees with b boundaries, n non-boundary vertices, k boundary vertices, and decorated with stems as follows: 1) each non-boundary vertex carries two stems, 2) for $1, \ldots, b$, the i-th boundary, of size called k_i, carries overall $k_i + 2$ stems [4]. We obtain:

Lemma 4.4 ([CFL10]) *For fixed $b > 0$, any tree in $\mathcal{P}^{(b)}_{n,k}$ can be encoded in a number $\ell(n,k)$ of bits that satisfies*

$$\ell(n,k) \sim 2k + \log_2 \binom{4n+2k}{n} \quad \text{as } n+k \to \infty.$$

The encoding of the tree is done by a contour word similarly as in [PS06]. Moreover, this encoding is asymptotically optimal with respect to n and k (when $n+k \to \infty$), since it matches the lower bound that one can derive from Brown's counting formula [Bro64].

4.3.3 *Encoding in higher genus*

For dealing with the higher genus case, it suffices to make some simple observations. First, as discussed in [McDo8] (Lemma 4.1), for any graph G on a surface S of genus $g > 0$ with n vertices, there exists a non-contractible cycle C on S such that C crosses G at vertices only, and $|G \cap C| \leqslant \sqrt{2n}$; C is in fact the cycle with smallest number of intersections and can be computed in time $O(n \log(n))$ for fixed genus g [Kuto6]. For a triangulation M on S with b boundaries (boundaries seen as faces), C can be deformed in each triangular face f to pass by one edge around f (but we do not deform C inside the boundary faces). After this, cut S along C; this yields a triangulation M' of genus $g-1$ with two *special* boundary-faces f_1, f_2 bounded each by C (indeed, cutting splits C into two copies), with otherwise at most $2b$ boundaries (because each of the b boundaries of M might be crossed by C, thus becoming two boundaries after cutting). We add a star into f_1 and f_2, so the boundaries are only the ones arising from the boundaries of M (the locations of the two special stars have be stored to recover M from M', which costs only $O(\log(n))$ in memory). If M has n non-boundary vertices and k boundary vertices, M' will have $n' \leqslant n + 2 + |C|$ non-boundary vertices and $k' \leqslant k + 2b + |C|$ boundary vertices, with $|C| \leqslant \sqrt{2(n+k)}$. By induction on g, M' can be encoded asymptotically optimally, i.e., with a word of length $\ell(n',k') \sim 2k' + \log_2 \binom{4n'+2k'}{n'}$. Since $\ell(n',k') \sim \ell(n,k)$ when $n+k \to \infty$ and when $n'+k' = n+k+O(\sqrt{n+k})$, and since only memory $O(\log(n))$ is necessary to recover M from M', the encoding in genus g is also asymptotically optimal.

Theorem 4.5 (Castelli Aleardi, Fusy, and Lewiner [CFL10]) *Given an orientable surface S of fixed topology $\tau = (g,b)$, it is possible to encode any triangulation on S having n inner vertices and k boundary vertices, such that the length $\ell(n,k)$ of the encoding word satisfies, as $n+k \longrightarrow \infty$:*

$$\ell(n,k) \sim \log_2 |\mathcal{T}^{(\tau)}_{n,k}|$$

where $\mathcal{T}^{(\tau)}_{n,k}$ denotes the set of triangulations on S with n inner vertices and k boundary vertices. Moreover, the encoding phase requires $O(n+k)$ time if $g = 0$ and $O((n+k)\log(n+k))$ time if $g > 0$, while decoding takes $O(n+k)$ time.

4 R satisfies this property since, in the contracted tree, the total outdegree $k_i + 3$ of the special vertex s_i consists of one outgoing edge to the father of s_i plus $k_i + 2$ outgoing stems

Data structure	size (references)	navigation	vertex access	vertex adjacency	dynamic updates
Edge-based data structures [GS85; Bau72; Bau75]	$18n + n$	$O(1)$	$O(1)$	$O(d)$	yes
triangle based [Boi+02]/Corner Table	$12n + n$	$O(1)$	$O(1)$	$O(d)$	yes
Directed edge [CKS98]	$12n + n$	$O(1)$	$O(1)$	$O(d)$	yes
Compact half-edge [AJ05]	$12n + n$	$O(1)$	$O(1)$	$O(d)$	yes
2D catalogs [CDM11]	$7.67n$	$O(1)$	$O(1)$	$O(d)$	yes
Star vertices [KT01]	$7n$	$O(d)$	$O(1)$	$O(d)$	no
TRIPOD [SS99] or our Thm 4.7	$6n$	$O(1)$	$O(d)$	$O(d)$	no
SOT [GR09]	$6n$	$O(1)$	$O(d)$	$O(d)$	no
ESQ [CADR12]	$4.8n$	$O(1)$	$O(d)$	$O(d)$	yes
Thm 4.8 (no vertex reordering)	$5n$	$O(1)$	$O(d)$	$O(d)$	no
Thm 4.8 (with vertex reordering)	$4n$	$O(1)$	$O(d)$	$O(d)$	no
Thm 4.9 (with vertex reordering)	$5n$	$O(1)$	$O(1)$	$O(1)$	no

Table 2: Comparison between existing data structures for triangle meshes. All storage and runtime bounds hold in the worst case. The degree of the accessed vertex is denoted d. Storage costs are expressed in terms the number of references (as a function of the number n of vertices).

4.4 COMPACT MESH DATA STRUCTURES: RELATED WORKS

A basic requirement for mesh data structures, in addition to representing the combinatorics and geometry of a 3D shape, is the *traversability*: the user should be provided with fast navigational operators allowing to perform a mesh traversal (such as walking around a vertex, or performing a DFS visit of the primal or dual graph). Sometimes the user may ask for additional functionalities such as the *indexability* (allowing to access in constant to each vertex or triangle given its index) or the *modifiability* (the manipulation of meshes may require to perform updates such as vertex insertions/deletions, edge collapses and edge flips). The existing representations which are commonly used in practice are easy to implement and provides fast navigational operators and dynamic local updates. But they have a drawback: in order to match these performances they store a lot of information, which makes the data structures quite redundant and not really suitable for representing very large meshes (Table 2 reports comparisons of mesh representations).

Compact data structures aim to reduce the connectivity cost, since connectivity largely exceeds the geometry cost. The typical cost of geometric coordinates ranges from $3 \times 16 = 48$ to $3 \times 32 = 96$ bits per vertex (bpv). While storing connectivity requires between $13 \times 32 = 416$ bpv and $19 \times 64 = 1216$ bpv (depending whether triangle, vertex indices are described with 32 or 64 bits per reference) adopting standard data structures [Bau75; Boi+02], as reported in Table 2.

PRACTICAL SOLUTIONS. An effective way of for dealing with huge meshes is to design *compact data structures* requiring small storage while still guaranteeing a fast and practical way of accessing mesh elements.

Succinct representations [CDS05a; CDS05b; CDS08], run under the *word-RAM model*, provide an optimal solution but are mainly of theoretical interest, since the amount of memory required in practice is quite important even for very large meshes. Some attempts to exploit the algorithmic framework of succinct representations in practice had lead to a space efficient dynamic data structure [CDM11]. The main idea is to gather together neighboring faces into small groups of triangles (called *patches*). While references are still of size $\Theta(\log n)$, grouping triangles allows us to save some references (corresponding to edges internal to a given patch). In particular a triangulation (possibly having handles and boundaries), can be represented using at most 7.67 references per vertex, while supporting $O(1)$ time navigation and local updates in $O(1)$ amortized time[CDM11] (a more concise solution [CADR12] requiring $4.8n$ references/vertex can be obtained through face/vertex reordering, slightly increasing the cost for updates).

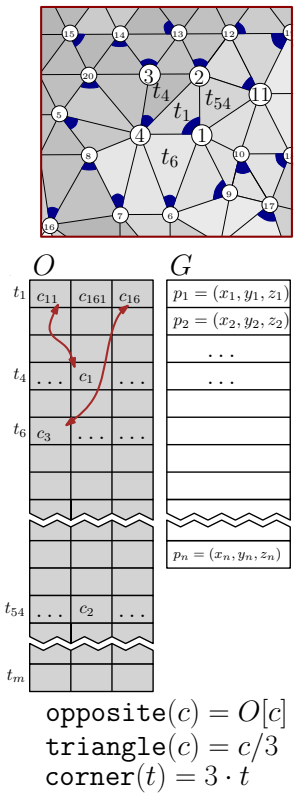


Figure 4.9: Illustration of the SOT data structure [GR09].

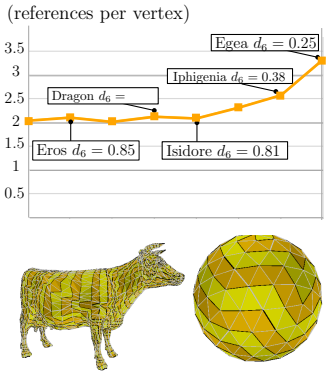


Figure 4.10: We have implemented the construction of the LR data structure [Gur+11a] and evaluated its compression performances: results are expressed in terms of references per vertex. Meshes are sorted from left to right according to d_6, the proportion of degree 6 vertices (regular meshes are listed leftmost). Our tests involve real-world meshes from the aim@shape repository and confirm the intuition that LR achieves good compression rates for regular meshes (with high value of d_6).

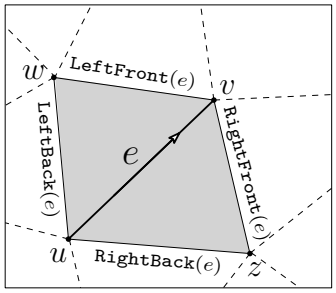
REDUCING REDUNDANCY THROUGH FACE REORDERING. One of the ingredients for reducing the storage requirements is to perform a reordering of the faces or the edges of the input mesh. The main idea used in the SOT data structure [GR09], illustrated in Fig. 4.9, is to implicitly represent the map from triangles to corners (`triangle` operator), and the map from corners to vertices (`vertex` operator), through a face reordering. First match each vertex to an incident triangle (in such a way a triangle is matched with at most one vertex). Then permute triangles in such a way that the triangle associate with the i-th vertex v_i has number i (thus the first n triangles appearing in this ordering are the ones associated with a vertex). The corners of a triangle are listed consecutively, and the first one corresponds to the vertex matched for the triangle. The incidence relations are stored in an array O (of size $3m$) having 3 entries per triangle: $O[i]$ stores the index of the corner opposite to c_i (which is matched to vertex v_i, for $i \leqslant m$). Corners operators are easily supported in $O(1)$ time performing arithmetic operations. The only operation that cannot be performed in constant time is vertex access: retrieving vertex v_i requires to walk around its incident faces until c_i is reached (v_i being matched to c_i).

Theorem 4.6 (Gurung and Rossignac [GR09]) *Given a triangulation (possibly having handles and boundaries), there exists a data structure using $6rpv$ which supports $O(1)$ time navigation (retrieving a vertex of degree d requires $O(d)$ time).*

MORE CONCISE HEURISTIC SOLUTIONS. Adopting some interesting heuristics one may obtain even more compact solutions [Gur+11b; Gur+11a; Gur+13], requiring better space requirements in practice, but with no theoretical guarantees in the worst case. For instance, the face pairing approach of the *SQUAD* data structure improves the SOT bounds to about $(4 + \varepsilon)$ references per vertex: as shown by experiments ε is usually a small value (between 0.09 and 0.3 for the tested meshes). Another heuristic combines the face and vertex re-ordering with computation of a nearly Hamiltonian ring spanning almost all vertices: as reported in [Gur+11a], the *LR* data structure is able to represent a triangulation using between 2.04 and 3.16 references per vertex in the case of regular meshes (we plot in Fig. 4.10 the experimental evaluation obtained with our implementation of the construction phase of the *LR* data structure).

4.5 OUR CONTRIBUTION: COMPACT DATA STRUCTURES FOR TRIANGULATIONS

NAVIGATIONAL INTERFACE. In order to design our compact data structures we have first to choose the right set operations that our solution will support. For our purposes we adopt the interface of the *winged-edge representation* [Bau72; CKS98], which offers navigational operators which are equivalent to the ones supported by the Half-edge representation. More precisely, given an edge $e = (u, v)$ oriented toward v, which is incident to (u, v, w) (its left triangle) and to (v, u, z) (its right triangle), the operators defined in Fig. 4.11 allows us to retrieve the four undirected edges $\{v, z\}$, $\{v, w\}$, $\{u, z\}$ and $\{v, w\}$, and perform full navigation in the mesh as required in geometric processing algorithms. For instance, their combination allows us to walk around the edges incident to a given face, or to iterate on the edges incident to a given vertex (as depicted in Figure 4.11).



$\texttt{LeftBack}(e) = \{u, w\}$
$\texttt{LeftFront}(e) = \{v, w\}$
$\texttt{RightBack}(e) = \{u, z\}$
$\texttt{RightFront}(e) = \{v, z\}$
$\texttt{Source}(e) = u$
$\texttt{Target}(e) = v$
$\texttt{Point}(u) = (u_x, u_y, u_z)$
$\texttt{Edge}(u)$ returns an edge outgoing from u

```
Degree(u) {
    // Enumerating edges incident to u
    e = Edge(u);
    f = e; d = 0;
    do{
        if (u ==Source(f))
            f =LeftBack(f) ;
        else
            f =RightFront(f) ;
        d = d + 1;
    } while f ≠ e;
    return d;
}
```

Figure 4.11: (left) Navigational operators supported by the *winged-edge representation* [Bau72]: these operators can be combined to perform mesh traversal, for example for enumerating the edges incident to a vertex (see pseudo-code on the right).

OVERVIEW OF OUR SOLUTION. In order to design new compact array-based data structures, we make use of two ingredients. The first idea is to exploit, as done in [SS99], the existence Schnyder woods which will allow storing only two references per edge (namely the `LeftFront` and `RightFront` incident edges). The second idea consists in performing, as done in [CKS98; GR09; Gur+11b], a reordering of cells (the edges in our case) to implicitly represent the map from vertices to edges, and the map from edges to vertices. More precisely, let us consider an input plane triangulation $\mathcal{G}$ endowed with a Schnyder wood (T_0, T_1, T_2), and define the tree $\overline{T}_0$ by adding edges (V_1, V_0) and (V_2, V_0) to T_0; after adding the edge (V_2, V_1) to the tree T_1 each edge gets a color and an orientation, as depicted in Figure 4.13.

In order to help the intuition we prefer to replace the three indices $\{0, 1, 2\}$ (operations are done modulo 3) with three labels $\{\texttt{red}, \texttt{blue}, \texttt{black}\}$ which are cyclically ordered using the operator $\texttt{next}$ (and $\texttt{prev}$) defined as: $\texttt{next}(\texttt{red}) = \texttt{blue}$, $\texttt{next}(\texttt{blue}) = \texttt{black}$, $\texttt{next}(\texttt{black}) = \texttt{red}$.

Since the local Schnyder condition ensures exactly one outgoing edge of each color (except for vertices V_0, V_1, V_2) an edge (u, v) can also be identified by its origin u and its color c. For each color c and each vertex u, we store two vertex indices $S_c^{\texttt{left}}[u]$ and $S_c^{\texttt{right}}[u]$ and two booleans $C_c^{\texttt{left}}[u]$ and $C_c^{\texttt{right}}[u]$ describing the source and color of the two neighboring edges of edge $u_c^{\nearrow}$ as detailed below and one boolean $I_c[u]$ indicating the existence of incoming edge at u of color c.

Vertices will be identified by integers $0 \leqslant i < n$; and an oriented edge (u, v) having source u and color c will be represented by the pair (u, c) (also denoted $u_c^{\nearrow}$).

4.5.1 *Our first solution: scheme description*

Combining Schnyder woods and the re-ordering of edges one can obtain an array-based representation using only 6 references: this first representation is still quite simple (and relatively easy to implement), so that we can shortly provide its complete description and sketch the main arguments underlying its correctness.

Our data structure consists of several arrays of size n:
- three arrays of booleans $I_{\texttt{red}}$, $I_{\texttt{blue}}$, and $I_{\texttt{black}}$ (I standing for *incoming*),
- six arrays of booleans $C^{\texttt{left}}_{\texttt{red}}$, $C^{\texttt{right}}_{\texttt{red}}$, $C^{\texttt{left}}_{\texttt{blue}}$, $C^{\texttt{right}}_{\texttt{blue}}$, $C^{\texttt{left}}_{\texttt{black}}$, $C^{\texttt{right}}_{\texttt{black}}$ (C standing for *color*),
- six arrays of vertex indices $S^{\texttt{left}}_{\texttt{red}}$, $S^{\texttt{right}}_{\texttt{red}}$, $S^{\texttt{left}}_{\texttt{blue}}$, $S^{\texttt{right}}_{\texttt{blue}}$, $S^{\texttt{left}}_{\texttt{black}}$, $S^{\texttt{right}}_{\texttt{black}}$ (S standing for *source*),
- an array P storing the geometric coordinates of the points.

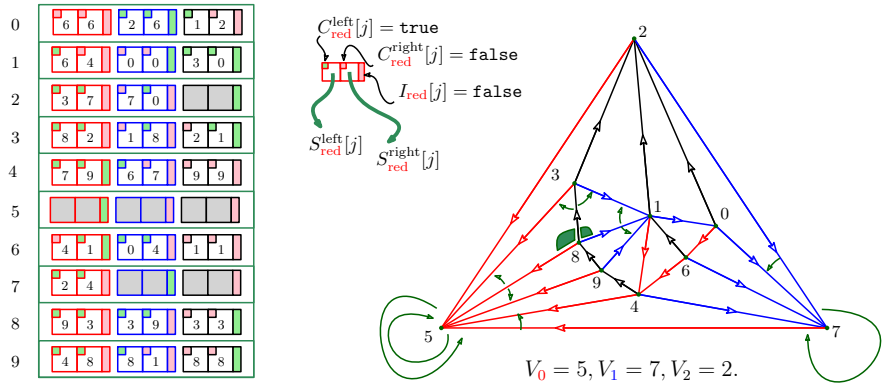


Figure 4.13: Illustration of the data structure described by Theorem 4.7. For each vertex we store 6 references in table S, corresponding to the source of the front neighbors of the 3 outgoing edges. Tables I, C, S are drawn as an array of size n where booleans are represented by colors (green=`true`, pink=`false`).

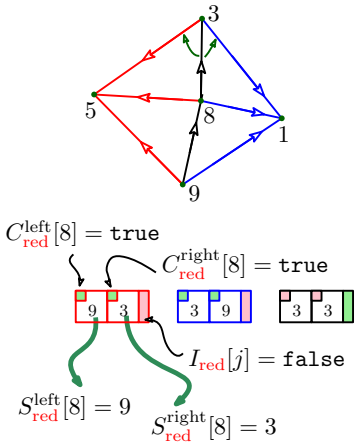


Figure 4.12: In th example above vertex 8 has no red incoming edges thus $I_{red}[8]$ is `false` (pink); the left front edge of the red edge outgoing from Vertex 8: $8^{\nearrow}_{red} = (8,5)$ is $(9,5) = 9^{\nearrow}_{red}$ thus the first box of line 8 of the array is green (same color) and contains 9.

These arrays store the following information (refer to Figure 4.13) for a vertex u and a color c (the three vertices of the outer face do not have all their outgoing edges and must obey special rules):

$$I_c[u] = \begin{cases} \texttt{true} & \text{if vertex } u \text{ has at least one incoming edge of color } c \\ \texttt{false} & \text{otherwise} \end{cases}$$

$$S_c^{\texttt{left}}[u] = \texttt{Source}(\texttt{LeftFront}(u_c^{\nearrow}))$$

$$S_c^{\texttt{right}}[u] = \texttt{Source}(\texttt{RightFront}(u_c^{\nearrow}))$$

Using the coloring rules, these two neighboring edges have only two possible colors: c and $\texttt{next}(c)$ (resp. $\texttt{prev}(c)$) for a left neighbor (resp. right neighbor). Arrays C_c store that information (see Fig. 4.12 for an illustration):

$$C_c^{\texttt{left}}[u] = \begin{cases} \texttt{true} & \text{if } \texttt{Color}(\texttt{LeftFront}(u_c^{\nearrow})) = c \\ \texttt{false} & \text{otherwise} \end{cases}$$

$$C_c^{\texttt{right}}[u] = \begin{cases} \texttt{true} & \text{if } \texttt{Color}(\texttt{RightFront}(u_c^{\nearrow})) = c \\ \texttt{false} & \text{otherwise} \end{cases}$$

Theorem 4.7 (Castelli Aleardi and Devillers [CD18]) *Let $\mathcal{T}$ be a plane triangulation with n vertices. The representation described above requires $6n$ references, while allowing support of* `Target` *operator in $O(d)$ time (when dealing with a degree d vertex) and all other operators in $O(1)$ worst case time.*

PROOF: Let e be an edge of color c, with source u and target v, whose incident left and right triangles are (u,v,w) and (v,u,z) respectively. The navigational operations supported by the Winged-edge data structure can be implemented as described below.

OPERATORS `Edge(u)` AND `Point(u)`: to get the index of an edge incident to a given vertex u we simply returns the edge $u^{\nearrow}_{red}$. The geometric coordinates of vertex u are naturally stored in $P[u]$.

OPERATOR $\texttt{Source}(e)$: is a trivial operation since $e = (u, v)$ is encoded as the pair (u, c) in our representation.

OPERATOR $\texttt{LeftFront}(e)$: by definition of arrays S and C, $\texttt{LeftFront}(e)$ is the edge of source $S^{\texttt{left}}_{\texttt{Color}(e)}[\texttt{Source}(e)]$ and color $\texttt{Color}(e)$ if the boolean value $C^{\texttt{left}}_{\texttt{Color}(e)}[\texttt{Source}(e)]$ is $\texttt{true}$, and color $next(\texttt{Color}(e))$ otherwise.

OPERATOR $\texttt{LeftBack}(e)$: We have to distinguish three cases, depending on the color of edges (u, w) and (v, w) (as illustrated in Figure 4.14):

Case 1: $I_{prev(\texttt{Color}(e))}[\texttt{Source}(e)]$ is false. This case is easy to handle, since (u, w) is the edge with source u and color $next(\texttt{Color}(e))$ (see Fig. 4.14(c), there is no incoming blue edge at u: $\texttt{LeftBack}(e)$ must be red and outgoing at u).

Case 2: $I_{prev(\texttt{Color}(e))}[\texttt{Source}(e)]$ is true and $C^{\texttt{left}}_{\texttt{Color}(e)}[\texttt{Source}(e)]$ is true. Then (w, v) is of color $\texttt{Color}(e)$ and (v, w) can be accessed as $\texttt{LeftFront}(e)$ and $\texttt{LeftBack}(e)$ is the edge with source $\texttt{Source}(\texttt{LeftFront}(e))$ and color $prev(\texttt{Color}(e))$ (there are incoming blue edge at u, thus $\texttt{LeftBack}(e)$ is blue. Since $\texttt{LeftFront}(e) = \{v, w\}$ is black, it is oriented from w to v and its source is also the searched source of $\texttt{LeftBack}(e)$).

Case 3: $I_{prev(\texttt{Color}(e))}[\texttt{Source}(e)]$ is true and $C^{\texttt{left}}_{\texttt{Color}(e)}[\texttt{Source}(e)]$ is false, then (v, w) is of color $next(\texttt{Color}(e))$ and $\texttt{LeftBack}(e)$ is the edge given by $\texttt{LeftFront}(\texttt{LeftFront}(e))$. As in Case 2 $\texttt{LeftBack}(e)$ is blue, but the edge $\texttt{LeftFront}(e) = \{v, w\}$ is red and oriented from v to w. $\texttt{LeftBack}(e)$ is then accessible as $\texttt{LeftFront}(\texttt{LeftFront}(e))$.

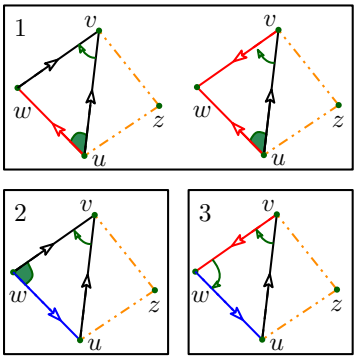


Figure 4.14: Case analysis of Theorem 4.7: in these pictures the edge (u, v) is assumed to be black.

OPERATOR $\texttt{Target}(e)$: unfortunately we have not stored enough information to return v in $O(1)$ time: the idea is to iteratively turn around vertex v (running, for example, the code given in Fig. 4.11), starting from edge (u, v) in cw direction (or ccw direction) until we get an edge $e' = (v, x)$ having v as source. Then compute $\texttt{Source}(e')$ in $O(1)$ time as above, which results in the target of (u, v). This procedures ends after at most $d - 3$ steps (for a vertex v of degree d), since each vertex has 3 outgoing edges.

OPERATORS $\texttt{RightBack}(e)$ AND $\texttt{RightFront}(e)$: observe that the traversal of the right face (u, v, z) incident to (u, v) can be handled in a similar manner as above. Operators $\texttt{RightBack}(e)$ and $\texttt{RightFront}(e)$ can be deduced by symmetry from operators $\texttt{LeftBack}(e)$ and $\texttt{LeftFront}(e)$, because of the symmetry of Schnyder woods and of our use of reference.

□

4.5.2 *Further reducing the space requirements*

In order to save one (or two) references per vertex, we make use of maximal Schnyder woods and we allow to permute input vertices (reordering the vertices according to a given permutation). One main ingredient is to make use of a breadth-first search traversal of $\overline{T}_0$, whose corresponding vertex ordering (denoted *cwBFS*) has the following useful property: the vertices having the same parent vertex u in $\overline{T}_0$ will have consecutive labels, when traversed turning cw around u from $u^{\nearrow}_{black}$ (see Figure 4.15). We then reorder all vertices (their associated data) according to their cwBFS label, and we store entries in table C accordingly. This allows us to save one reference per vertex: we do not store a reference to $\texttt{RightFront}$ for edges in $\overline{T}_0$, which

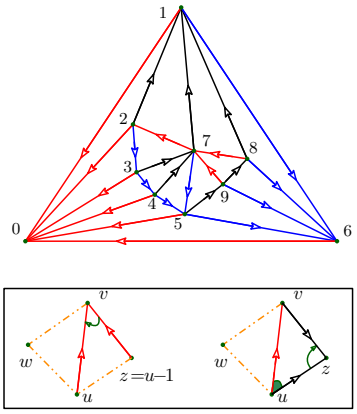


Figure 4.15: A planar triangulation (endowed with its maximal Schnyder wood) whose vertices are labeled according to the cwBFS traversal of tree $\overline{T}_0$ (top). Bottom pictures illustrate two cases involved in the proof of Theorem 4.8: we now store only one reference for red edges, since most adjacency relations between edges in $\overline{T}_0$ are implicitly described by the cwBFS labels.

leads us to not store tables C^{right}_{red} and S^{right}_{red}. We retrieve the corresponding information using the ordering of vertices as explained below (it is even possible to save one more reference, but this implies to change the information involving black edges: the details are omitted here, please refer to [CD18]).

Theorem 4.8 (Castelli Aleardi and Devillers [CD18]) *Let $\mathcal{T}$ be a plane triangulation with n vertices endowed with its maximal Schnyder wood. Then $\mathcal{T}$ can be represented using $5n$ references, while supporting navigation as in Theorem 4.7. If one is allowed permuting the input vertices (their associated geometric data) then $\mathcal{T}$ can be represented using $4n$ references, with the same runtime complexity.*

The proof of the result above is based on a case analysis which is much more involved than the one of Theorem 4.8 and tedious to follow (omitted here). Two interesting cases are illustrated by the pictures in Figure 4.15. Compared to the representation of Theorem 4.7 we lose the information concerning `RightFront` for red edges (which was previously explicitly stored in C^{right}_{red} and S^{right}_{red}): an important remark is that, given a red edge $e = (u, v)$, we can still retrieve its right siblings in $\overline{T}_0$ just using its cwBFS label. If (z, v) is oriented toward v (thus red as (u, v)), its source is simply $z = u - 1$. Otherwise, we exploit the assumption that the Schnyder wood is maximal: since clockwise oriented triangles are forbidden, the edge $\{z, u\} = \texttt{RightFront}(u^{\nearrow}_{red})$ must be black, and in this case we can retrieve it with a memory access, first computing the edge (u, z), which is also black and oriented toward z.

4.5.3 *Additional features*

ALL OPERATIONS IN $O(1)$ WORST CASE TIME. We can further exploit the redundancy in our representation in order to improve the computational cost of the navigation: all navigational operations, including checking the adjacency between neighboring vertices, can be supported in worst case constant time slightly increasing the space requirements as stated below (the details are omitted here and can be found in [CD18]):

Theorem 4.9 ([CD18]) *If one is allowed permuting input points, then there exists a compact representation requiring $5n$ references which supports all navigation operators (including `Target` and `Adjacent`) in worst case $O(1)$ time.*

DECODING THE TRIANGULATION FROM COMPRESSED FORMAT. A main issue common to many compact data structures [CADR12; CDM11; Gur+11b; GR09; Gur+11a], is that an explicit representation of the entire mesh must be kept in main memory during the whole construction phase: this is needed, for example, to process vertices and edges (or faces), which must be renumbered according to a prescribed mesh traversal. This preliminary construction phase can greatly increase the overall memory requirements, especially for very huge meshes: in addition to the space storage of the compact data structure to construct (between $4n$ and $8n$ references for most compact representations), one has to use between $19n$ references for an explicit representation (such as *Winged-edge* or *Half-edge*), and a few additional memory references for the implementation of the mesh processing (typically, a graph traversal). To address this issue, we can take advantage of the compressed format for triangle meshes described in Section 4.2, which makes use of Schnyder woods in order to reduce the storage requirements.

More precisely, it turns out that it is possible to save our array-based compact representations in a compressed format so that we can reconstruct on the fly our structure in linear time, without any extra memory cost and in a streamable fashion.

The first construction of our representation still needs an explicit representation, in particular for the computation of the Schnyder wood.

As shown in [CD18] the decoding procedure is in two steps and consists in performing a linear scan of the two words W, W' encoding respectively the red tree T_0 and the incoming black degree of the vertices.

In a first pass, the linear scan of the W word allows the construction of the red tree by inserting the vertices (numbered according to the ccw-depth-traversal) into a stack: in this way we fill all red columns of array C and S. In parallel, we can read the word W' which allows us to fill the black columns of array C and S. No extra memory is required for an explicit storage of the red stack: as the blue columns are not involved during this first step, one blue column can be used to provide an array-based implementation of such a stack. When considering the red and black trees together, one obtains a planar map where only blue edges are missing.

A second pass is required for retrieving the information about blue edges. In the case where one is provided with a complete representation of such a map (as in [HKL99; BB09]) retrieving the location of blue edges is straightforward: the source vertex of a blue edge is known by applying the local Schnyder rule (the coloring and orientation of edges around a vertex), and edge destinations can be recovered by performing a facial walk of each red/black face. In our case such this solution cannot be applied: after the first decoding step the red and black trees are recovered, but only a partial knowledge of the red/black map is available.

The strategy for retrieving blue edges is not trivial and requires some technical details which are omitted here (see [CD18] for a more detailed explanation).

So we have to apply a new strategy, slightly different from the one used in Section 4.2, which consists in iteratively discovering and visiting in cw order the blue edges incident to a given vertex x (this procedure is performed independently for each vertex).

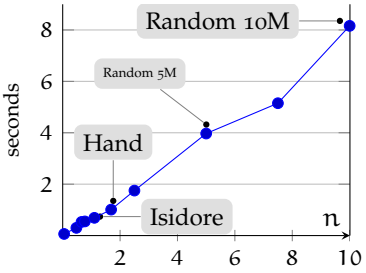
STREAMABLE ENCODING SCHEME. As described above, the first decoding phase (construction of red and black trees), requires a parallel reading of the two black and red words, or to perform a linear scan of the whole encoding in two passes: to first construct the red tree, and then to recover black edges (with a second linear scan). In practice, this can be a limitation in the case of streaming applications. In order to avoid this problem, and to make our data structure fully streamable, we can slightly modify our encoding scheme, by interleaving the symbols in the red and black words.

More precisely, the bits in the red and black words can be mixed in a single binary word as follows. We start with the encoding of the red tree, where (and) symbols are replaced by 0 and 1 bits respectively. Let us assume we are visiting and encoding a vertex v: just after the 0_v bit corresponding to the first visit of v, we encode the black in-degree of v by writing a block consisting of d 1 bits followed by a 0 bit. The mixed code corresponding to the example of Figure 4.4 is given below

$$0_0\,0_0\,0_1\,0_1\,1_1\,0_2\,0_2\,1_2\,0_3\,0_3\,1_3\,0_4\,0_4\,1_4\,0_5\,0_5\,0_6\,110_6\,0_7\,10_7\,1_7\,0_8\,10_8\,1_8\,1_6\,1_5\,0_9\,1110_9 1_9\,1_0.$$

where we provide colors and subscripts just to help intuition. As detailed in [CD18], it is possible to distinguish between red and black symbols, since during the decoding phase we perform a linear scan of this mixed word, by taking into account the Schnyder local rule.

Building *Half-edge DS* from OFF



Decoding compressed format

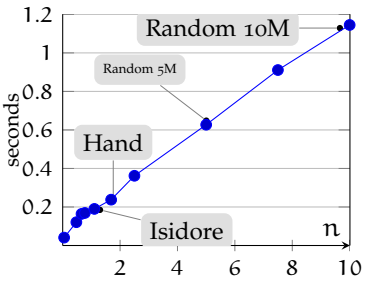


Figure 4.16: Comparison of the pre-processing runtime costs. The top chart reports the timing cost for building the half-edge data structure from a binary OFF file: we allocate 6GB of RAM memory for running the whole phase. The bottom chart corresponds to the whole decoding phase of our *CDS6n* data structure from the binary compressed format described in Section 4.2: in this case we only allocate 800MB of RAM for the JVM. Timings are expressed in seconds as a function of the size of the input mesh (millions of vertices) and represent the average time over several runs of our tests. All results are taken from [CD18].

4.6 EXPERIMENTAL RESULTS

4.6.1 *Preprocessing: construction vs. decoding.*

We first evaluate the runtimes of our data structures concerning the pre-processing construction phase. The charts in Fig. 4.16 show the construction costs (expressed in seconds) of our data structure using 6n references (referred to as *CDS6n*, Thm 4.7) compared to the pre-processing phase involving the Half-edge data structure. The reported runtimes confirm the asymptotic linear time behaviour of all steps of our algorithms.

CONSTRUCTING COMPACT DATA STRUCTURES FROM OFF FORMAT. The runtimes reported in the top chart of Fig. 4.16 correspond to the construction phase of the Half-edge data structure from an input binary OFF file, storing both the geometry (3D vertex locations) and the mesh connectivity (the mesh is stored with a so-called *shared vertex representation*, where each face is represented with the indices of the three incident vertices).

Once we have in main memory the half-edge representation of the input mesh we can run the incremental algorithm to endow the triangulation with a Schnyder wood orientation and build arrays underlying our compact data structures. The overall timing cost is dominated by the construction of the Half-edge representation in main memory: this is the unique bottleneck, in terms of memory requirements, of our pre-processing phase (in order to run this phase on the tested meshes we have to allocate up to 6GB of RAM memory). It is worth noting that most compact data structures [GR09; Gur+11b; Gur+11a; Gur+13; CADR12] have the same limitation.

MORE EFFICIENT CONSTRUCTION: DECODING THE COMPRESSED FORMAT. One main advantage of our encoding/decoding algorithm sketched in Section 4.5.3 is that the input mesh can be directly constructed from a compressed input file, without using additional memory for an intermediate representation (such as the Half-edge data structure) and without computing the Schnyder wood, which leads in overall to a smaller construction cost. The bottom chart of Fig. 4.16 reports the runtimes of the decoding from the compressed format described in Section 4.2 (the cost includes the scan of the input file also storing the geometry information): our procedure[5] is extremely fast and can be performed using small memory resources even for large meshes (only 800MB of RAM for the JVM are sufficient for the tested meshes).

4.6.2 *Mesh navigation: runtime performances*

In order to evaluate the performance of our representations, we have implemented a Java array-based version of the *Winged-edge* data structure (requiring 19 references per vertex): the plots in Fig. 4.17 report comparisons with our compact data representations.

As in previous works [CKS98; GR09; Gur+11b; Gur+11a; Gur+13; CADR12] we consider two usual processing procedures: computing *vertex degrees* (involving edge navigation) and *vertex normals* (involving vertex access operators and geometric calculations). In all tests we allocate enough memory so that all representations of the tested 3d models fit in main memory; vertices are accessed sequentially according to their original order in the input

5 A more detailed discussion of the implementation and performances can be found in [CD18].

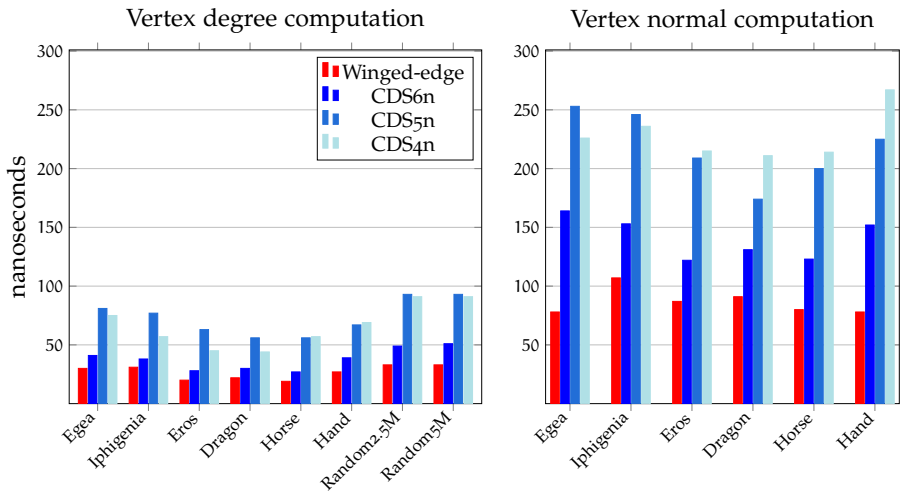


Figure 4.17: Comparison of runtime performances for the computation of vertex degrees and vertex normals. Our compact data structures are compared to our implementation of the *Winged-edge* data structure. All results are expressed in nanoseconds per vertex and represent the average over hundred of executions.

mesh (except for the data structure using $4n$ references of Thm 4.8, where vertices are re-ordered according to the cw-BFS traversal of the tree T_0). We have written implementations of our data structure described in Thm 4.7 (referred to as *CDS6n*) and of the more compact versions described in Thm 4.8 (called *CDS5n* and *CDS4n* respectively).

The implementation details are omitted here: a more comprehensive discussion on the encoding of service bits and the use of bit shifts and masks for manipulating integer references can be found in [CD18].

PRACTICAL INTEREST OF COMPACT DATA STRUCTURES. As one could expect, non-compact mesh representations (e.g. *Winged-edge* data structure) are faster in general: refer to the runtimes plotted in Fig. 4.17. In overall, our data structures achieve good trade-offs between space usage and runtimes. While being between 3.17 and up to 4.75 times more compact for connectivity than Winged-edge, our structures are slightly slower: they lose in average on the tested meshes a factor 1.33 for *CDS6n*, 2.65 for *CDS5n* and 2.31 for *CDS4n*, when performing vertex degree computations.

Our representations are still competitive when considering the `target` operator, which requires $O(d)$ time in the worst case for a vertex of degree d for our compact data structures (observe that the `target` operator requires $O(1)$ time in the case of *Winged-edge*). The results reported in the right chart of Figure 4.17 provide a comparison of the runtimes for the vertex normal computation, and show that our representations are slower than other data structures by a factor between 1.5 and 2.5 in average (all geometric calculations involve simple float precision). The higher cost of the `target` operator for the *CDS6n*, *CDS5n* and *CDS4n* representations is compensated by the cost of geometric calculations, which dominates the runtimes, and is the same for all representations.

5 DRAWING GRAPHS ON SURFACES

Most graphs we encounter in computer science, discrete mathematics and real-life applications are not random but exhibit some strong structural properties and regularities. The main goal of graph drawing algorithms is to compute a layout which is easily readable and helps the user to recover the structure of the underlying graph. This problem is very challenging, one of the main reasons being that real-world graphs are really big and difficult to inspect: thus we need deep mathematical tools to provide guarantees on the quality of the result and efficient algorithmic tools to obtain fast practical implementations.

Most works tend to take profit of some assumptions on the input graph: for example the graph class to which it belongs or some properties, such as some notion of connectedness or some structural parameter. In particular graph planarity have played a crucial role in the domain of Graph Drawing since its early years, and led to the discover of beautiful mathematical theorems and deep algorithmic tools, whose interest and relevance went well beyond the original visualization purposes.

While there is a large literature on planar drawings, the problem of drawing graphs on surfaces has been addressed less frequently from both the theoretical and algorithmic point of view. This chapter focuses on my contributions [1] to the problem of efficiently drawing graphs on surfaces, and provides a short overview of the main recent advances in this domain.

5.1 GRAPH DRAWING IN THE PLANE ("AS I HAVE KNOWN IT")

When dealing with planar (or locally planar) graphs a very basic problem addressed by the Graph Drawing community consists in mapping the vertices and edges of a graph onto a region in the plane or a portion of a 3D surface. Most of the time edges are represented as smooth curves (very often as straight-line segments), and the drawing is required to be *crossing-free*. Sometimes the drawing is asked to satisfy some further aesthetic criteria, in order to obtain a pleasing and readable result. For example, one could seek for *good vertex resolution* for ensuring that vertices are not too close to one another.

FORCE-DIRECTED METHODS. In the planar case, an elegant solution to the graph drawing problem is provided by the so-called *barycentric embedding*, one of Tutte's masterpieces [Tut63]. In his pioneering work Tutte showed how to compute vertex positions by solving a system of linear equations: the method applies to a 3-connected planar graph and the resulting drawing is guaranteed to be crossing-free, also allowing to fix the positions of outer vertices (which are mapped to the vertices of a given convex polygon). The solution can be also reformulated in terms of a system of springs converging to an equilibrium position, and has inspired a huge number of force-directed embedding algorithms (the interested reader can find a

1 The contributions of this chapter are originated from the collaborations with Eric Fusy and Olivier Devillers [CDF18b], together with our students Gaspard Denis [CDF18a] and Anatolii Kostrygin [CFK14].

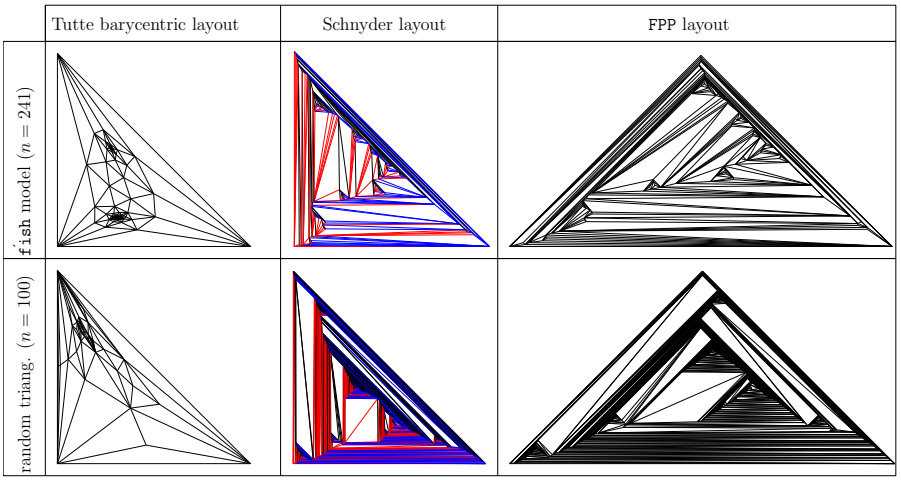


Figure 5.2: Comparison of planar layouts (all layouts are obtained with our Java implementations): the Tutte embedding is compared to the FPP algorithm and the Schnyder drawing. The pictures above show the layouts of a planar random triangulation of size $n = 100$ and a triangle mesh (the fish graph, top pictures). For the sake of readability the FPP layout is re-scaled (its grid size is originally $2n \times n$).

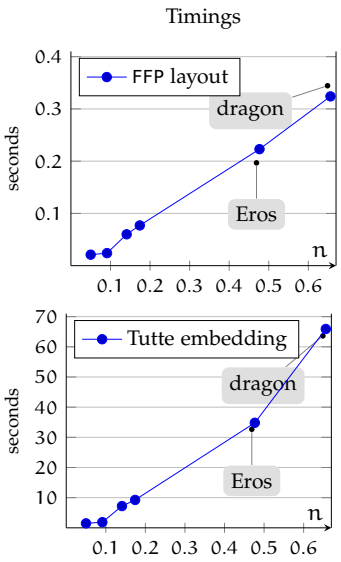


Figure 5.1: Comparison of runtime performances. Timings are expressed in seconds as a function of the size (millions of vertices). The runtimes for the FPP layouts are obtained with our Java implementation where we make use of our reformulation of the shift principle (as described in Section 5.3). For the solution of the linear equations underlying the Tutte embedding, we make use of the Java implementation of the conjugate gradient solver provided by the Matrix Toolkit Java (MTJ) library.

comprehensive survey of force-directed methods in [Kob13]). These methods allows us to obtain very nice layouts achieving several desirable aesthetic criteria, such as uniform edge lengths, low angle distortion or even the preservation of symmetries (see top pictures in Fig. 5.2).

A main common drawback is that one cannot achieve a good vertex resolution, since geometric computations (involving for example linear solvers) lead to vertex coordinates of exponential size. Moreover, they are quite expensive in terms of runtime costs, since their implementation requires iterative linear solvers (for dealing with large sparse matrices) or sometimes non-linear optimization methods, making these approaches slower and less robust than combinatorial graph drawing tools.

COMBINATORIAL ALGORITHMS. A solution to the problems above is provided by the so-called *straight-line grid drawings* [FPP90; Kan96; Sch90]: the graph is embedded on a regular grid whose area is typically polynomial with respect to the size of the graph. A further advantage of this approach is that the combinatorial algorithms perform essentially arithmetic computations on integers of bounded magnitude (no roundings are needed).

For the planar case, many classes of algorithms have been proposed to solve this task: in this document we focus on the *face-counting* approach proposed by Schnyder [Sch90] (whose principle is illustrated in the bottom pictures of Figure 5.3) and on the *shift* algorithm described by Fraysseix, Pach, and Pollack [FPP90] (whose steps are sketched in the top pictures of Figure 5.3), both achieving very good vertex resolution (quadratic area) and time complexity (running in linear time in the worst case). Their practical performances are extremely good compared to force-directed methods: using combinatorial algorithms one can deal in real-time with very large graphs, being able of processing several millions of vertices per second. A comparison between the timing costs is given in Figure 5.1.

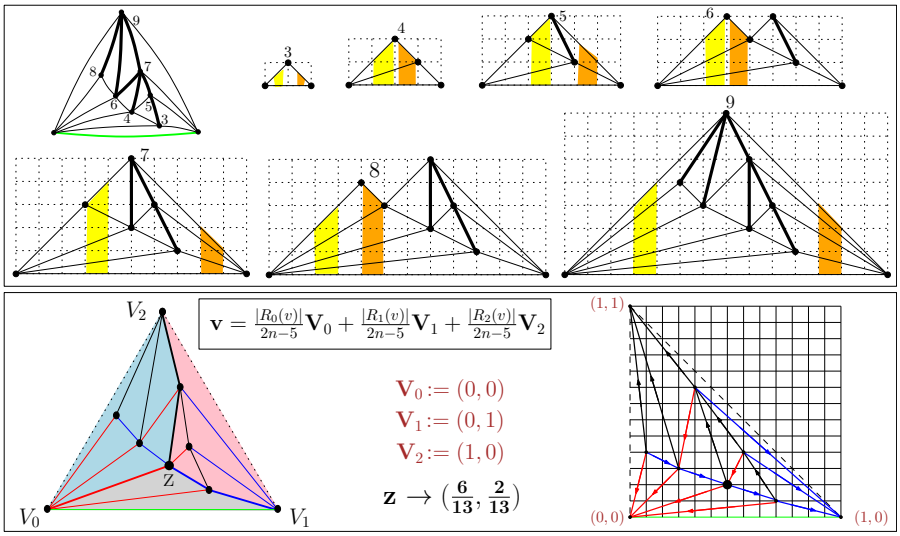


Figure 5.3: Top pictures show a planar triangulation endowed with a canonical ordering and the $n-2$ steps of the FPP algorithm [FPP90] that embeds a graph with n vertices on a grid of size $2n \times n$. This algorithm relies on a *shift* principle: we proceed incrementally by adding vertices according to the canonical ordering. At each step the drawing is *stretched* horizontally by adding two vertical bands of width 1, and the new vertex is placed on the outer face at the intersection of two rays having slopes $+1$ and -1. A detailed description of the shift principle will be provided later in this chapter: note that the FPP layout can be recovered as a special case using our algorithm for cylindrical triangulations presented in Section 5.3. (Bottom) The Schnyder drawing relies on a face-counting principle: the location of a given inner vertex v is defined in terms of barycentric coordinates with respect to the outer vertices: more precisely, v is located at the barycentric combination $\sum_{i\in\{0,1,2\}} \frac{|R_i(v)|}{2n-5}\mathbf{V_i}$, where $|R_i(v)|$ is the number of faces belonging to the region $R_i(v)$ (as defined in Section 2.2.1) and $2n-5$ is the total number of inner faces.

Unfortunately, the resulting layouts are rather unpleasing and fail to achieve some basic aesthetic criteria that help readability: they often have long edges and large clusters of tiny triangles, as illustrated by the pictures in Fig. 5.2.

5.2 DRAWINGS HIGHER GENUS GRAPHS: RELATED WORKS

While there is a large literature on planar layouts, the problem of drawing graphs on surfaces has been addressed less frequently from both the theoretical and algorithmic point of view.

DRAWINGS HIGHER GENUS GRAPHS: LAYOUT SETTING. For drawing higher genus graphs most existing works [WKS01; GGT06; Moh96; MR98; Cha+12; DGK11; GL14; Zit94] consider the planar (periodic) setting: for instance, in the case of toroidal maps, the main goal is to obtain crossing-free layouts on the flat torus.

Assume from now on that w and h are positive integers. For a cylindrical map G, a *periodic straight-line drawing* of G of width w and height h is a crossing-free straight-line drawing (edges are drawn as segments, two edges can meet only at common end-points) of G on the flat cylinder of width w and height h, such that the vertex-coordinates are in $\mathbb{Z}/w\mathbb{Z} \times [0..h]$ (i.e.coordinates are integers). Similarly, for a toroidal map G, a *periodic*

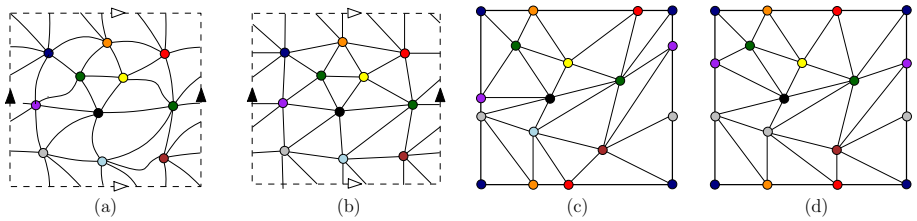


Figure 5.4: (a) A toroidal triangulation G; (b) a periodic straight-line drawing of G which is not straight-frame; (c) a non-periodic straight-frame drawing of G (where G is planarly unfolded using a cut-graph); (d) a periodic straight-frame drawing of G.

straight-line drawing of G of width w and height h is a crossing-free straight-line drawing of G on the flat torus of width w and height h, such that the vertex-coordinates are in $\mathbb{Z}/w\mathbb{Z} \times \mathbb{Z}/h\mathbb{Z}$.

Sometimes the layout is required to satisfy some stronger aesthetic constraints: a *straight-frame* drawing of a toroidal graph G is a planar straight-line drawing of G such that the outer face contour is an axis-aligned rectangle, corresponding to a canonical polygonal scheme of G (the graph is cut along fundamental cycles, whose vertices and edges are placed on horizontal and vertical lines respectively).

Pictures (b) and (d) of Fig. 5.4 show two periodic straight-line drawings of the same graph. Pictures (c) and (d) of Fig. 5.4 correspond to straight-frame drawings: the example in (b) corresponds to a non-periodic drawing since only the relative order of the vertices on the outer face is preserved (vertices consecutive on a fundamental cycle are also consecutive on the boundary).

5.2.1 *Grid-drawings of toroidal graphs: combinatorial algorithms*

While here we focus on combinatorial algorithms we should point out that some of the numerical algorithms for graph drawing can be used, or easily extended, to deal with higher genus graphs (for instance Kobourov and Wampler [KW05] show how to make use of force-directed layouts in the non-euclidean setting).

FACE-COUNTING APPROACH. A recent and elegant method for drawing toroidal graphs with polynomial grid size is the algorithm of Gonçalves and Lévêque [GL14], which is an adaptation of Schnyder's drawing principles to the torus and relies on the properties of crossing Schnyder woods (defined in Section 3.4). It achieves both the periodicity requirement and polynomial grid-size: more precisely, the size of the periodic regular grid is $O(n^2) \times O(n^2)$ for simple toroidal triangulations (no loops or multiple edges) and is $O(n^4) \times O(n^4)$ for essentially simple (simple in the periodic representation) toroidal triangulations. Gonçalves and Lévêque also extend these ideas to 3-connected toroidal maps (similarly, Schnyder woods for plane triangulations have been extended to plane 3-connected maps [Felo1]), but as opposed to the planar case [Felo1], the periodic drawings of 3-connected toroidal maps they obtain do not necessarily have the desired convexity property.

`fpp` LAYOUT OF THE PLANARIZED GRAPH. Recently some methods for the straight-line planar drawing of genus g graphs with polynomial grid area $O(n^3)$ in the worst case have been described [Cha+12; DGK11]. Such methods unfold the graph in the plane along a *cut-graph* and then apply the incremental approach of the `FPP` algorithm to obtain a straight-frame drawing of the graph. However, these methods do not yield easily periodic representations: for example, in the case of a torus, the boundary vertices might not be aligned, so that the drawing does not give rise to a periodic drawing.

Nevertheless the decomposition underlying this approach is interesting and inspired our works on toroidal and spherical drawings (see Sections 5.4 and 5.5), where we can overcome the alignment problem.

5.3 OUR CONTRIBUTION: PERIODIC TOROIDAL DRAWINGS

In order to obtain periodic drawings of toroidal maps, we first compute a pair of parallel non-contractible cycles defined a so-called tambourine (as described in Section 2.1.2): the removal of the tambourine leads to a cylindrical map with two inner and outer boundaries, denoted Γ_{inn} and Γ_{ext}. The first problem to address is thus to get a periodic straight-line drawings of a cylindrical map: the solution for the triangulated case is detailed below.

5.3.1 *Drawing cylindrical simple triangulations (with no chord at Γ_{inn}).*

Here we provide a detailed description of the algorithm for dealing with a cylindrical simple triangulation G with no chord at Γ_{inn} (the more general case, allowing chords at Γ_{inn}, as well as multiple edges and loops will be sketched later).

We first compute a canonical ordering of G as described in Section 3.3, and then draw G in an incremental way. We start with a cylinder of width $2|\Gamma_{inn}|$ and height 0 (i.e., a circle of length $2|\Gamma_{inn}|$) and draw the vertices of Γ_{inn} equally spaced on the circle: space 2 between two consecutive vertices[2].

Then the strategy for each $k \geqslant 1$ is to compute the drawing of G_k out of the drawing of G_{k-1}. Note that the set of vertices on the boundary cycle of G_{k-1}, denoted B_{k-1}, that are neighbours of v_k forms a path γ on B_{k-1}. Traversing γ with the outer face of G_{k-1} to the left, let e_ℓ be the first edge of γ and e_r be the last edge of γ (note that $e_\ell = e_r$ if v_k has only two neighbours on B_{k-1}). Let also a_k be the starting vertex and let b_k be the ending vertex of γ. Two cases can occur.

(1) If, in the drawing of G_{k-1} obtained so far, $\text{slope}(e_\ell) < 1$ and $\text{slope}(e_r) > -1$, then we can directly insert v_k in the drawing. We place v_k at the intersection of the ray of slope 1 starting from a_k and the ray of slope -1 starting from b_k, and we connect v_k to all vertices of γ by segments.

(2) If $\text{slope}(e_\ell) = 1$ or $\text{slope}(e_r) = -1$, then we cannot directly insert v_k as done in Case (1), because the edges e_ℓ and $\{a_k, v_k\}$ would overlap if $\text{slope}(e_\ell) = 1$, or the edges e_r and $\{b_k, v_k\}$ would overlap if $\text{slope}(e_r) = -1$. We first have to perform stretching operations (thereby increasing the cylinder width by 2) to make the slopes of e_ℓ and e_r smaller than 1 in absolute value. Define the *x-span* of an edge e in the cylindrical drawing as the number of columns $[i, i+1] \times [0, +\infty)$ that meet the interior of e (we have no need for a more complicated definition since, in our drawings, a column will never meet an edge more than once). Consider the dual forest F^* for the canonical ordering restricted to G_{k-1} (recall the its definition given in Section 3.3). Let P_ℓ (resp. P_r) be the root-path of e^*_ℓ (resp. e^*_r) in F^*. We stretch the cylinder by inserting a vertical strip of length 1 along P_ℓ and another along P_r, see Fig. 5.5. This comes down to increasing by 1 the x-span of each edge of G_{k-1} dual to an edge in P_ℓ, and then increasing by 1 the x-span of each edge dual to an edge in P_r (note that P_ℓ and P_r are not necessarily disjoint, in which case the x-span of an edge dual to an edge in

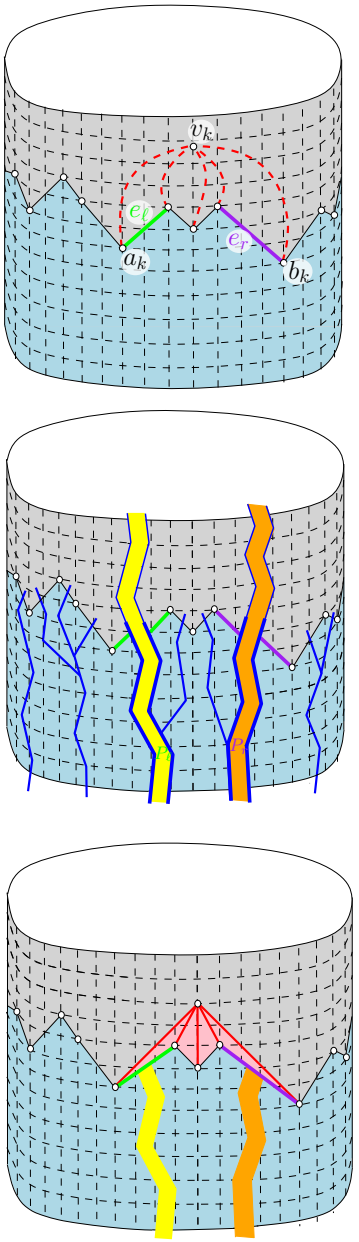


Figure 5.5: One step of the incremental algorithm for drawing cylindrical triangulations. Two vertical strips of width 1 (each one along a path in the dual forest) are inserted in order to make the slopes of e_ℓ and e_r smaller than 1 in absolute value. Then the new vertex and its edges connected to the upper boundary can be drawn in a planar way.

2 It is also possible to start with any configuration of points on a circle such that any two consecutive vertices are at even distance.

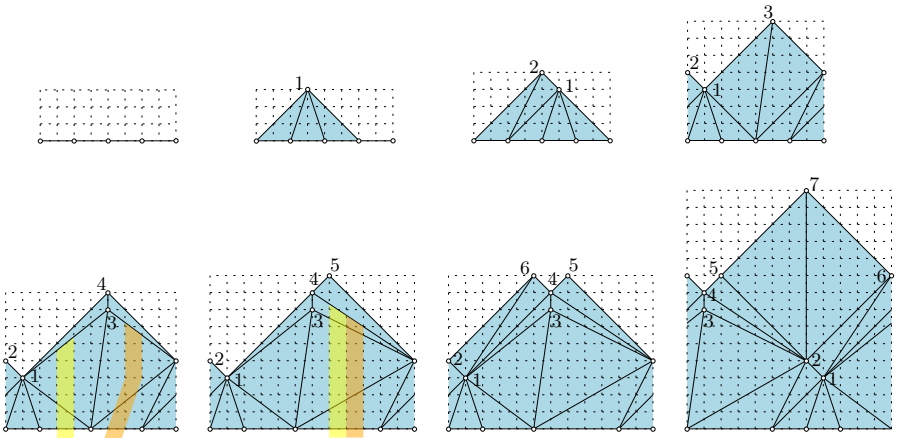


Figure 5.6: Complete execution of the algorithm computing an x-periodic drawing of a cylindrical simple triangulation with no chordal edge at B_{inn}. The vertices are processed in increasing label (the canonical ordering is the one computed in Fig. 3.13).

$P_\ell \cap P_r$ is increased by 2). After these stretching operations [3], whose effect is to make the slopes of e_ℓ and e_r strictly smaller than 1 in absolute value, we insert, as in Case (1), the vertex v_k at the intersection of the ray of slope 1 starting from a_k and the ray of slope -1 starting from b_k, and we connect v_k to all vertices of γ by segments.

Note that in the two cases (1) and (2), the two rays from a_k and b_k actually intersect at a grid point since the Manhattan distance between any two vertices on B_{k-1} is even. A complete execution of this algorithm is illustrated in Fig. 5.6.

One can easily check that the drawing remains crossing-free: this relies on the fact that all edges of the upper boundary have slope at most 1 in absolute value, and on the following inductive property (similar to the one used in [FPP90]), which is maintained at each step k from 1 to n:

> **Pl**: for each edge e on B_k (the upper boundary of G_k), let P_e be the root-path of e^* in F^*, let E_e be the set of edges dual to edges in P_e, and let δ_e be any nonnegative integer. Then the drawing remains crossing-free after successively increasing by δ_e the x-span of all edges of E_e, for all $e \in B_k$.

BOUNDS ON THE GRID-SIZE. Let us denote by w the width and by h the height of the cylinder on which G is drawn. If $|\Gamma_{inn}| = t$ then the initial cylinder is $2t \times 0$; and at each vertex insertion, the grid-width grows by 0 or 2. Hence $w \leqslant 2n$. In addition, due to the slope conditions (slopes of boundary-edges are at most 1 in absolute value), the vertical span of every edge e is not larger than the current width at the time when e is inserted in the drawing. Hence, if we denote by v the vertex of Γ_{ext} that is closest (at distance d) from Γ_{inn}, then the ordinate of v is at most $d \cdot (2n)$. And due

3 In the FPP algorithm for planar triangulations, the step to make the (absolute value of) slopes of e_ℓ and e_r smaller than 1 is formulated as a shift of certain subgraphs described in terms of the underlying forest F. The extension of this formulation to the cylinder would be quite cumbersome. We find the alternative formulation with strip insertions more convenient for the cylinder. In addition it also gives rise to a very easy linear-time algorithm, whose practical performances are extremely good, as confirmed by the running times reported in Fig. 5.1. Another linear-time version of the FPP algorithm is given in [CP95]).

to the slope conditions, the vertical span of Γ_{ext} in the drawing is at most $w/2 \leqslant n$. Hence the grid-height is at most $n(2d+1)$.

LINEAR-TIME COMPLEXITY. An important remark is that, instead of computing the x-coordinates and y-coordinates of vertices in the drawing, one can compute the y-coordinates of vertices and the x-span of edges (as well as the knowledge of which extremity of the edge is the left-end vertex and which extremity is the right-end vertex). In a first pass, for k from 1 to n, one computes the y-coordinates of vertices and the x-span r_e of each edge $e \in G$ at the time $t = k$ when it appears on G_k (as well one gets to know which extremity of e is the left-end vertex). Afterwards if $e \notin F$, the x-span of e might further increase due to insertion of new vertices; let s_e be the total further increase undergone by e. Note that for each edge e not in F, if $e \notin \Gamma_{ext}$ there is a certain step k such that $e \in B_{k-1}$ and $e \notin B_k$. Let $w_e \in \{0,1,2\}$ be defined as the stretch (increase of x-span) that e undergoes just before adding v_k to the drawing; in case $e \in \Gamma_{ext}$ no such step k exists and we assign $w_e = 0$. We call w_e the *weight* of e (the quantities w_e can be computed in a first pass, together with the quantities r_e). Let P be the root-path of e^*. When stretching e just before adding v_k, all edges dual to edges of P undergo the same stretch, by w_e. In other words, if we denote by T^*_e the subtree of F^* hanging from e^* (including e^*), and denote by W_e the total weight of the dual of the edges in T^*_e, then $s_e = W_e$. Hence the total x-span of each edge $e \in G$ is given by $r_e + s_e$, where $s_e = 0$ if $e \in F$ or $e \in \Gamma_{ext}$, and $s_e = W_e$ if $e \notin F$ and $e \notin \Gamma_{ext}$. Since all quantities s_e can easily be computed in linear time from the quantities w_e, starting from the leaves and going up to the roots of F^*, this gives a linear-time algorithm.

To sum up, for the case of simple cylindrical triangulations with no chord at Γ_{inn} (and with no loops and no multiple edges) we can state the following:

Proposition 5.1 ([CDF18b]) *Given a simple cylindrical triangulation G with no chordal edge at Γ_{inn}, one can compute in linear time a crossing-free straight-line drawing of G on an x-periodic regular grid $\mathbb{Z}/w\mathbb{Z} \times [0..h]$ where —with n the number of vertices of G and d the edge-distance between the two boundaries— $w \leqslant 2n$ and $h \leqslant n(2d+1)$, such that the upper boundary is a broken line monotone in x formed by segments of slope in $\{+1,-1,0\}$ and the lower boundary is an horizontal line.*

Remark 1 *Note that our algorithm can be seen as an extension of the FPP algorithm, which works for simple planar quasi-triangulations, i.e., simple graphs embedded in the plane with triangular inner faces and a polygonal outer face (see Fig. 5.7).*

Remark 2 *For each edge e of Γ_{inn}, let r_e be the initial horizontal stretch of e in the drawing procedure (an even number, classically $r_e = 2$ to have a compact drawing). And let t_e be the final horizontal stretch in the drawing procedure. The vectors $R = (r_e)_{e\in\Gamma_{inn}}$ and $T = (t_e)_{e\in\Gamma_{inn}}$ are called* initial-stretch *and* final-stretch *vectors relative to the drawing of G (endowed with a given canonical ordering). Then the vector $S := T - R$ is an invariant, it does not depend on R since $s_e = t_e - r_e$ just depends on the underlying forest and dual forest given by the canonical ordering. Hence, if with R as initial stretch-vector we obtain a drawing with final-stretch vector T, then for any vector T' of the form $T + 2V$ —with V a vector of non-negative integers— we can obtain a drawing with final-stretch vector T', by taking $R' = R + 2V$ as initial-stretch vector instead of R.*

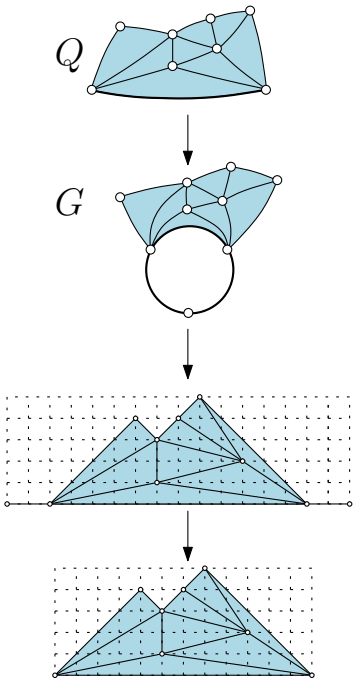


Figure 5.7: The FPP algorithm for simple planar quasi-triangulations can be recovered from our algorithm. Given a simple planar quasi-triangulation Q, we can turn it into a cylindrical simple triangulation G by adding a vertex of degree 2 connected to the two ends of the root-edge. Then the FPP drawing of Q is recovered from the periodic drawing of Q upon deleting the added vertex.

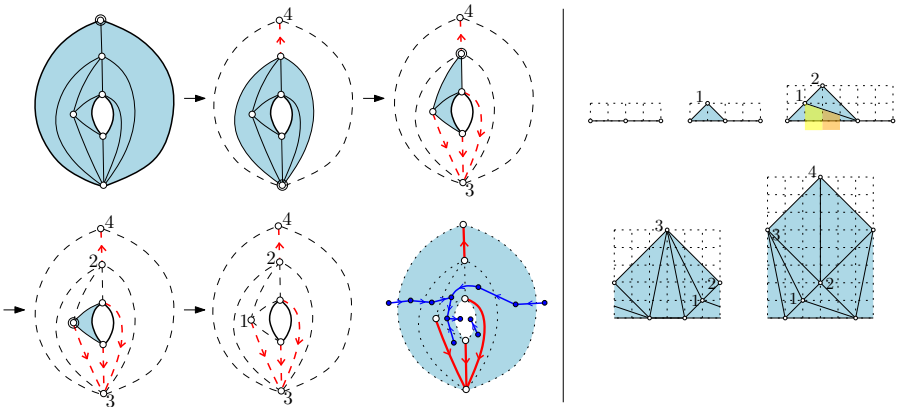


Figure 5.9: Left-side: the shelling procedure for an essentially simple loopless cylindrical triangulation G with no chord at Γ_{inn}; the last drawing shows the underlying forest and dual forest. Right-side: the incremental drawing algorithm.

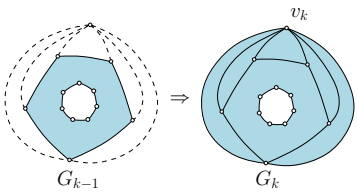


Figure 5.8: When 2-cycles are allowed, the additional case shown here can occur for the transition from G_{k-1} to G_k.

ALLOWING FOR NON-CONTRACTIBLE 2-CYCLES. Our method based on a canonical ordering and incremental drawing algorithm can be easily extended to deal with an essentially simple cylindrical triangulation G, having no loops but possibly with non-contractible 2-cycles. The definition of canonical ordering for G is exactly the same as for simple cylindrical triangulations, adding the possibility that G_k is obtained from G_{k-1} as shown in Fig. 5.8. Such a canonical ordering can be computed by a shelling procedure that extends the one of Section 3.3. A 2-cycle is called *internal* if its two incident vertices are not both on the outer boundary. This time, a vertex on the outer boundary and not on the inner boundary is called *free* if it is not incident to a chord nor incident to an internal 2-cycle.

The shelling procedure consists in choosing a free vertex at each step, and deleting it together with its incident edges, until there just remains the inner boundary and the cylindrical map is reduced to a cycle. (the argument that shows the existence of free vertices is omitted here, and a detailed presentation is given in [CDF18b]).

A linear time algorithm is also readily obtained by maintaining, for each outer vertex, how many neighbours on Γ_{ext} it has and how many internal 2-chords it is incident to. Note that such a canonical ordering also induces an underlying forest F and the dual forest F^*. These can be computed on the fly during the shelling procedure, in the same way as for simple cylindrical triangulations. Finally, the incremental drawing algorithm (and linear complexity using the dual forest) works exactly in the same way as for cylindrical simple triangulations. An example is shown in Fig. 5.9. The grid bounds are also the same as for simple triangulations (the arguments to obtain the bounds in the simple case did not use the fact that there are no 2-cycles). So this gives Proposition 5.1 for essentially simple triangulations with no loops. Finally, note that Remark 2 still holds here (the arguments are the same).

ALLOWING FOR NON-CONTRACTIBLE LOOPS. We finally explain how to deal with non-contractible loops. Our strategy is not to extend the notion of canonical ordering but simply to decompose (at the loops, which are nested) such a cylindrical map into a "tower" of components, where the only loops in each component are at the boundary-faces. Let G be an essentially simple

cylindrical triangulation with n vertices and at least one loop. There are a few cases to consider:

(a) Γ_{inn} *is the unique loop of* G. In that case, the algorithm of Section 3.3 (canonical ordering, shelling procedure, and incremental drawing procedure) works in the same way, see Fig. 5.10 for an example. Let $2m$ be the width of the drawing. By the arguments of Remark 2, for any $m' \geqslant m$, G has a periodic drawing of width $2m'$ and height at most $m'(2d+1)$, with d the edge-distance between the two boundaries.

(b) Γ_{ext} *is a loop,* Γ_{inn} *is possibly a loop, and there are no other loops.* Assume Γ_{ext} is a loop and G is not reduced to that loop (i.e., $\Gamma_{ext} \neq \Gamma_{inn}$). Let u be the vertex incident to the loop (note that u is not on Γ_{inn} since there is no chord at Γ_{inn}), and let c be the innermost 2-cycle incident to u. Cutting along c (see Fig. 5.11), we obtain two components: a planar triangulation T and a cylindrical essentially simple triangulation G' such that: G' has no loop except possibly at Γ_{inn}, G' has outer degree 2 and u is a free vertex for G'. Hence there is a canonical ordering for G' such that u is the first shelled vertex. Take a periodic drawing of G' for this canonical ordering and take an FPP drawing of T. The widths of the respective drawings are even numbers, denoted $2n_1$ and $2n_2$. Let $m = \max(n_1, n_2)$. By the arguments of Remark 2 it is possible to redraw the graph that has the smaller grid-width so that it gets width $2m$, after which both drawings are of width $2m$. Then the drawing of T (taken upside down) fits into the upper boundary of G' yielding a periodic drawing of G of width $2m$, see Fig. 5.11. The height of the drawing is at most $2dm \leqslant 2dn$, where d is the edge-distance between the two boundaries (indeed, in the usual bound $h \leqslant (2d+1)m$, the $+1$ in the parenthesis is due to the vertical extension of the upper boundary, which is 0 here).

(c) General case. Here we assume there is at least one loop. Let $\ell_1, \ldots, \ell_r$ be the sequence of nested loops of G, with ℓ_1 the innermost loop and ℓ_r the outermost loop; and let $G^{(0)}, \ldots, G^{(r)}$ be the $r+1$ components that result from cutting successively along all these loops. For $i \in [0..r]$ let d_i be the edge-distance between the two boundaries in $G^{(i)}$; and let d be the edge-distance between the two boundaries of G. Note that $d = \sum_i d_i$. Each component G_i has loops only at the boundary-face contours, hence has a periodic drawing (according to cases (a) and (b)) such that boundaries that are loops are drawn as horizontal lines. Let $2n_1, \ldots, 2n_r$ be the widths of the drawings of $G^{(1)}, \ldots, G^{(r)}$ thus obtained. Let $m = \max(n_1, \ldots, n_r)$. By the arguments of Remark 2, each of the graphs $G^{(i)}$ can be redrawn so as to have width $2m$. Stacking up all these drawings we obtain a periodic drawing of G of width $2m$, see Fig. 5.12. Regarding the grid size, the width is $2m$, with clearly $m \leqslant n$, and the height of the drawing of G_i is at most $2md_i$ for $i \in [0..r-1]$ and at most $m(2d_r+1)$ for $i = r$. Hence the total height is at most $m(2d+1) \leqslant n(2d+1)$. This establishes Proposition 5.1.

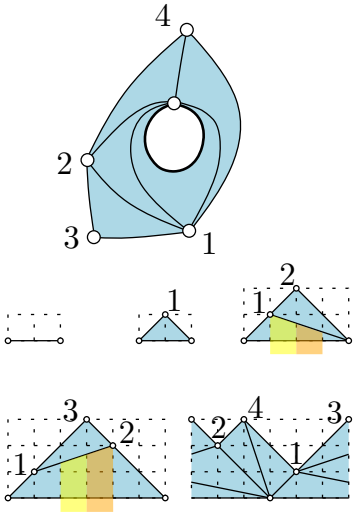


Figure 5.10: Drawing algorithm when the inner boundary is the unique loop.

5.3.2 *Drawing cylindrical triangulations having chords at* Γ_{inn}.

We finally explain how to draw a cylindrical essentially simple triangulation when allowing for chords incident to Γ_{inn}. It is good to view B_{ext} as the top boundary-face and B_{inn} as the bottom boundary face, and imagine a standing cylinder. For each chord e at the cycle Γ_{inn}, the *component under* e, denoted Q_e, is the face-connected part of G that lies below e; such a component is a quasi-triangulation (polygonal outer face, triangular inner faces)

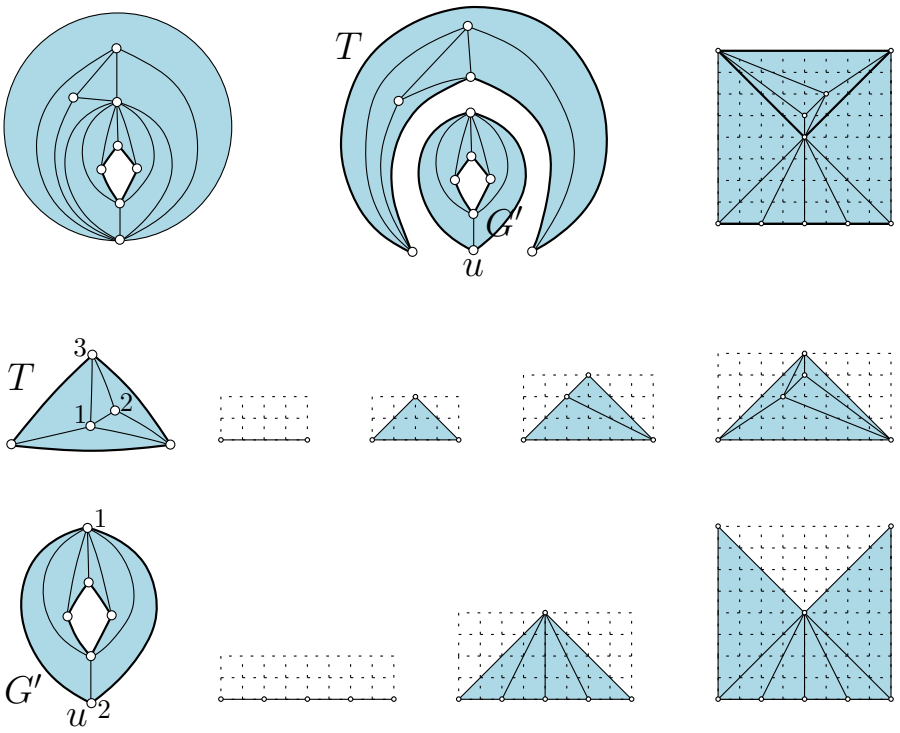


Figure 5.11: Drawing algorithm when the outer face contour is a loop (and there is no other loop except possibly at Γ_{inn}): the map is split along the innermost 2-cycle (the initial x-span of T is taken to be 4 instead of 2 so that the drawings of Q and G′ fit together).

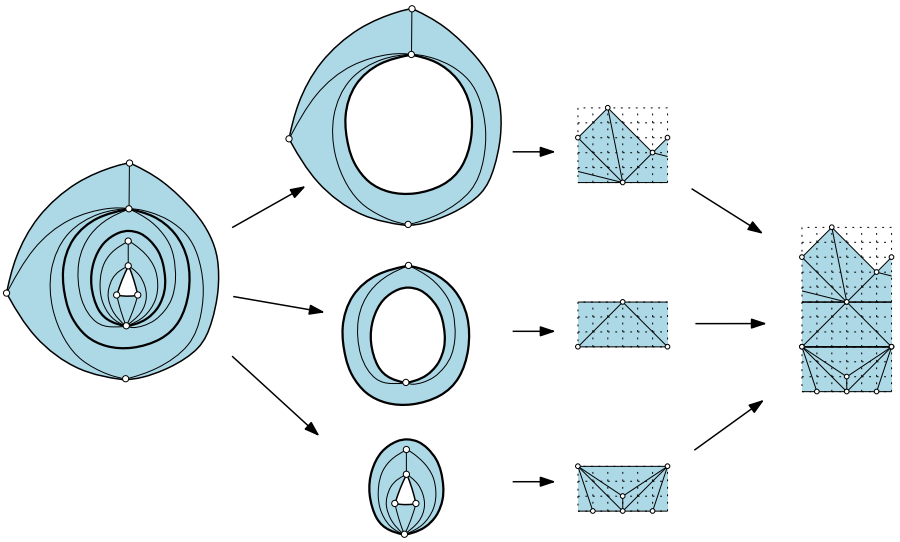


Figure 5.12: Drawing an essentially simple triangulation G (no chord at Γ_{inn}); G is first decomposed at its loops; each component is drawn so that the component-drawings have the same width, and can be stacked up to obtain a periodic drawing of G.

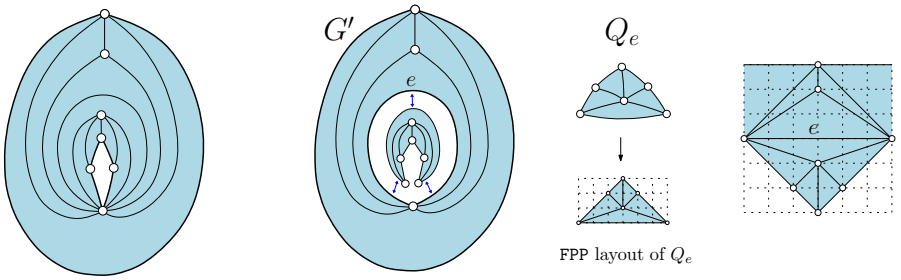


Figure 5.13: Drawing a cylindrical triangulation with chords at Γ_{inn} (essentially simple with loops and 2-cycles in the annular representation). To make enough space to place the component under e, one takes 4 (instead of 2) as the initial x-span of e.

rooted at the edge e. A chordal edge e of Γ_{inn} is *maximal* if the component Q_e under e is not strictly included in the component under another chord at Γ_{inn}. The *FPP-size* $|e|$ of such an edge e is defined as the width of the FPP drawing of Q_e. If we delete the component under each maximal chordal edge (i.e., delete everything from the component except for the chordal edge itself) we get a new bottom cycle C'_0 that is chordless, so we can draw the reduced cylindrical triangulation G' as explained in the previous section. Let w_e be the width of each edge e of C'_0 in this drawing. According to Remark 2, we can redraw G' such that each edge $e \in C'_0$ that is chordal in G has width $\ell(e)$, with $\ell(e)$ defined as the smallest integer that is at least $\max(w_e, |e|)$ and such that $\ell(e) - w_e$ is even (note that $\ell(e) \leqslant \max(w_e, |e| + 1)$).

Then for each maximal chord e of C_0, we draw the component Q_e under e using the FPP algorithm. This drawing has width $|e|$, with e as horizontal bottom edge of length $|e|$ and with the other outer edges of slopes in ± 1. We shift the left-extremity of e to the left so that the drawing of Q_e gets width $\ell(e)$, then we rotate the drawing of Q_e by 180 degrees and plug it into the drawing of G', see Fig. 5.13. The overall drawing of G is clearly planar and achieves good vertex resolution, as expressed by the result below, which generalizes Proposition 5.1 (the analysis of the grid-size is omitted here; more details can be found in [CDF18b]).

Theorem 5.2 ([CDF18b]) *For each essentially simple cylindrical triangulation G, one can compute in linear time a crossing-free straight-line drawing of G on an x-periodic regular grid $\mathbb{Z}/w\mathbb{Z} \times [0..h]$ where —with n the number of vertices of G and d the edge-distance between the two boundaries— $w \leqslant 2n$ and $h \leqslant n(2d+1)$, such that the upper boundary is a broken line monotone in x formed by segments of slope in $\{+1, -1, 0\}$ and the lower boundary is an horizontal line.*

5.3.3 *Periodic drawings on the torus*

Combining the results described in previous section with the computation of a tambourine, we can obtain a linear-time algorithm for computing periodic straight-line drawings of toroidal graphs. Our main result, stated in the triangulated case, is the following:

Theorem 5.3 *For each essentially simple toroidal triangulation $\mathcal{T}$, one can compute in linear time a crossing-free straight-line drawing of $\mathcal{T}$ on a periodic regular grid $\mathbb{Z}/w\mathbb{Z} \times \mathbb{Z}/h\mathbb{Z}$, where —with n the number of vertices and e_w the edge-width— $w \leqslant 2n$ and $h \leqslant 1 + 2n(e_w + 1) = O(n^{3/2})$.*

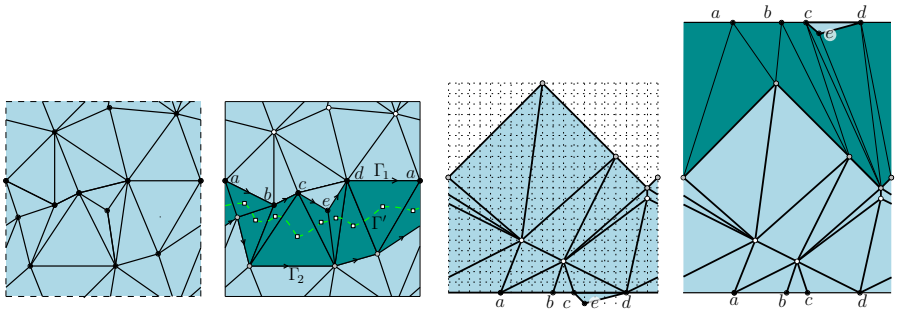


Figure 5.14: The main steps for drawing of a toroidal triangulation: first remove the edges inside a tambourine; then draw the obtained cylindrical triangulation, and insert the edges of the tambourine back into the drawing.

The fact that the grid size is $w \times h = O(n^{5/2})$ relies on an upper bound on the edge-width proved in [AH78] (namely $e_w \leqslant \sqrt{2n}$).

DRAWING TOROIDAL TRIANGULATIONS: OVERVIEW OF THE PROOF. Let $\mathcal{T}$ be an essentially simple toroidal triangulation with n vertices, and let Γ_1, Γ_2 be a tambourine of $\mathcal{T}$ (as defined in Section 2.1.2). By deleting the edges that are strictly inside the tambourine, one obtains a cylindrical triangulation G with Γ_1 the outer boundary and Γ_2 the inner boundary.

We can apply the drawing algorithm of Theorem 5.3 to obtain a periodic drawing of G on an x-periodic grid $\mathbb{Z}/w\mathbb{Z} \times [0..h]$. If we augment the height h of the drawing to $h' = h + w + 1$, and then wrap the x-periodic grid $\mathbb{Z}/w\mathbb{Z} \times [0..h]$ into a periodic grid $\mathbb{Z}/w\mathbb{Z} \times \mathbb{Z}/h'\mathbb{Z}$, and finally insert the edges inside the tambourine as segments (we insert the edges in the tambourine T in the unique way such that, looking from bottom to top, at least one edge in T goes strictly to the right, and all edges going strictly to the right have x-span at most w; in this way it is easy to check that the x-span of all edges in T is at most w). Then the slope properties —edges on Γ_1 and Γ_2 have slope at most 1 in absolute value while edges inside the tambourine have slopes greater than 1 in absolute value— ensure that the resulting drawing is crossing-free. Note that it is actually enough to augment the height by the least value such that the edges of T have slope greater than 1 in absolute value. See Fig. 5.14 for an example.

We now argue that we can find a tambourine Γ_1, Γ_2 so that the distance between the two boundaries Γ_1 and Γ_2 (in G) is smaller than the edge-width of $\mathcal{T}$; and we can find it *without having to compute* a curve $\Gamma_{\min}$ realizing the edge-width (this is crucial to obtain a linear-time complexity, since it is not known how to find such a curve $\Gamma_{\min}$ in linear time). Indeed, let $\{\Gamma_a, \Gamma_b\}$ be a basis of non-contractible cycles of $\mathcal{T}$ (computable in linear time, using for instance a cut-graph). Then at least one of Γ_a or Γ_b is not homotopic (parallel) to $\Gamma_{\min}$. Let Γ be one among $\{\Gamma_a, \Gamma_b\}$ that is not homotopic to $\Gamma_{\min}$, and let Γ_1, Γ_2 be a (computable) tambourine parallel to Γ. Since we are on the torus, $\Gamma_{\min}$ has to cross the tambourine Γ_1, Γ_2. Hence, the distance between the boundary-cycles (after deleting edges in the tambourine Γ_1, Γ_2) is smaller than the edge-width. In other words, if we choose the one cycle among $\{\Gamma_a, \Gamma_b\}$ that yields the smaller distance between the two boundaries of G, then this distance d is smaller than the edge-width e_w of $\mathcal{T}$. The grid-size of the drawing of G satisfies $w \leqslant 2n$ and $h \leqslant 2n(d+1)$, and the grid-size of the drawing of $\mathcal{T}$ is w, h' where $h' \leqslant h + w + 1 \leqslant 2n(d+1) + 2n + 1 =$

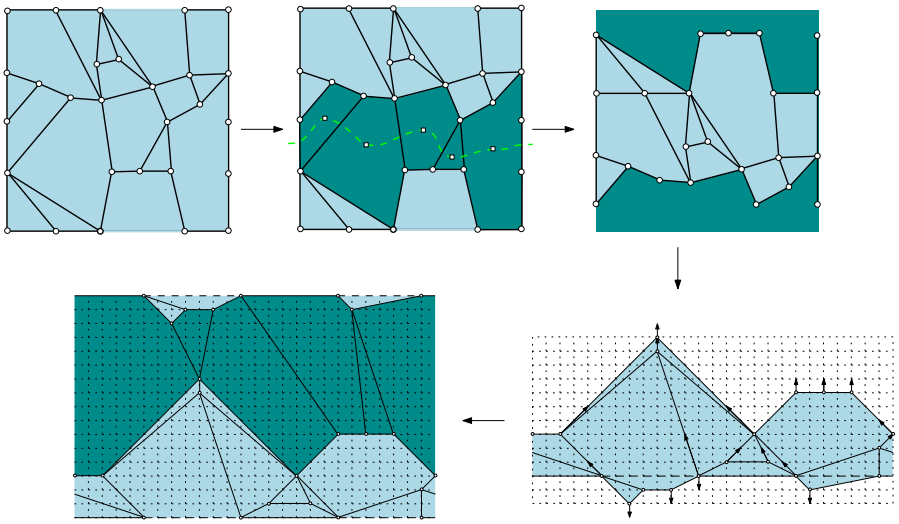


Figure 5.15: Drawing essentially 3-connected toroidal maps: first remove the edges inside a tambourine, and then draw the obtained (essentially) internally 3-connected cylindrical map; at the end insert the edges of the tambourine back into the drawing.

$1 + 2n(d+2)$. Since $d < e_w$ we conclude that the grid-height of the drawing of $\mathcal{T}$ is bounded by $1 + 2n(e_w + 1)$, as stated in Theorem 5.3.

DRAWING ESSENTIALLY 3-CONNECTED GRAPHS It turns out that these kind of technics described above for triangulations are also suitable to deal with the more general case of essentially 3-connected toroidal maps: the key idea is to make use of our generalization of the notion of canonical orderings for 3-connected planar graphs and the related grid drawing algorithm by [Kan96] (the proof is omitted here and a detailed discussion can be found in [CDF18b]).

The validity of our construction also relies on the fact that given an essentially 3-connected toroidal map $\mathcal{M}$ and a non-contractible cycle Γ, it is possible to compute in linear time a tambourine Γ_1, Γ_2 of $\mathcal{M}$ whose two cycles are homotopic to Γ. The proof of this fact, that extends the result for triangulations mentioned in Section 2.1.2, can be found in [CDF18b].

Our main result, illustrated by the pictures in Fig. 5.15, is stated below:

A periodic straight-line drawing on the flat torus is said to be weakly convex *if all corners have angle at most π.*

Theorem 5.4 (Castelli Aleardi, Devillers, and Fusy [CDF18b]) *For each essentially 3-connected toroidal map $\mathcal{M}$, one can compute in linear time a weakly convex crossing-free straight-line drawing of $\mathcal{M}$ on a periodic regular grid $\mathbb{Z}/w\mathbb{Z} \times \mathbb{Z}/h\mathbb{Z}$, where —with n the number of vertices and c the face-width— $w \leqslant 2n$ and $h \leqslant 1 + 2n(c+1)$. Since $c \leqslant \sqrt{2n}$, the grid area is $O(n^{5/2})$.*

The bound on the grid-size relies on the fact that we can triangulate an essentially simple toroidal graph $\mathcal{M}$, by adding edges but not adding vertices, so as to obtain an essentially simple toroidal triangulation $\tilde{\mathcal{M}}$. As already mentioned (and shown in [AH78]), $\tilde{\mathcal{M}}$ has a non-contractible cycle γ of length at most $\sqrt{2n}$: observe that this cycle can be locally deformed into a proper non-contractible curve meeting $\mathcal{M}$ at the vertices on γ.

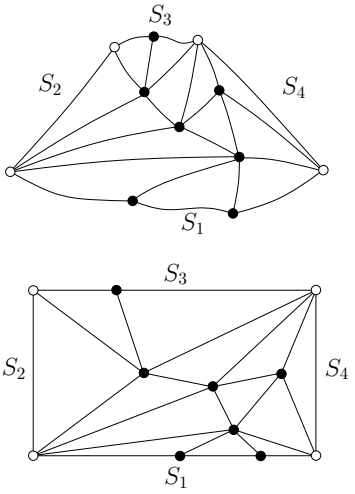


Figure 5.16: A 4-scheme triangulation G (corners are white) and a straight-frame drawing of G (bottom). G is not balanced, as $|S_1| \neq |S_3|$.

5.4 OUR CONTRIBUTION: PERIODIC STRAIGHT-FRAME DRAWINGS

We now address the problem of computing toroidal drawings adding a further constraint: we aim at obtaining a planar straight-frame drawing of a toroidal triangulation, for which the edges are not allowed to cross the boundary of the flat torus [4].

We follow the strategy adopted in [DGK11] for the toroidal case based on the following fact: any simple toroidal triangulation $\mathcal{T}$ can be cut along a subgraph (cut-graph) such that the resulting unfolded graph G is naturally a quasi-triangulation (referred to as 4-*scheme triangulation* and shortly denoted *4ST*) with 4 marked outer vertices, called *corners*, such that each path of the outer face contour between two consecutive corners is chordless. We will denote by $S_1, \ldots, S_4$ these outer paths (in clockwise order around the outer face), and denote by $|S_i|$ the length of S_i. Observe that cutting a toroidal triangulation in this way yields a *balanced* 4-scheme triangulation, having the additional property that $|S_1| = |S_3|$ and $|S_2| = |S_4|$.

So, our initial problem (drawing a toroidal triangulation) translates into the problem of computing a *straight-frame periodic drawing* of a (balanced) 4-scheme triangulation G (refer to Fig. 5.16 for an illustration): the abscissas of vertices of the same rank along S_1 and S_3 (ordered from left to right) should coincide, and the ordinates of vertices of the same rank along S_2 and S_4 (ordered from bottom to top) should also coincide. When G arises from a toroidal triangulation $\mathcal{T}$, this is exactly the condition to satisfy so that the drawing lifts to a periodic representation of $\mathcal{T}$ on the flat torus.

In this section we provide a short overview of the steps leading to our main result stated below (its validity relies on several technical lemmas whose proofs are omitted here: more details can be found in [CFK14]):

Theorem 5.5 (Castelli Aleardi, Fusy, and Kostrygin [CFK14]) *Each balanced 4-scheme-triangulation admits a periodic straight-frame drawing on a regular grid of size* $O(n^4 \times n^4)$. *The drawing can be computed in linear time.*

Our algorithm makes use of a new decomposition strategy for 4-scheme triangulations: we cut the triangulation into two components A, B along a certain path "close to" S_3, the "lower part" B is drawn using the algorithm of Duncan et al. [DGK11] specially adapted for our purpose (to make the ordinates of matching vertices in the lower parts of S_2 and S_4 coincide), and then we draw the "upper part" A with a suitably fixed outer frame (this is the most technical part, which requires a further decomposition of A into several pieces).

5.4.1 *Related works: description of the algorithm of Duncan et al. [DGK11]*

Define a *half-4-scheme triangulation*, shortly written H4ST, as a graph H embedded in the plane satisfying all conditions of a 4ST, except that the paths S_2 and S_4 are allowed to be empty, and the path S_3 is allowed to have chords and to meet S_1; S_1 is called the *bottom-path* of H. Given a 4ST G, an important ingredient in [DGK11] is the *river lemma*, which guarantees the existence of a so-called *river*, that is, a path P in the dual graph of G, such that the first edge of P crosses S_1, the last edge of P crosses S_3, and the two components

4 Observe that the Tutte's spring embedding algorithm gives a solution, but with exponential grid size, and the two combinatorial algorithms mentioned in this chapter [CDF18b; GL14] yield periodic grid-drawings for toroidal triangulations having the aesthetic disadvantage that some edges are allowed to cross the boundary of the drawing.

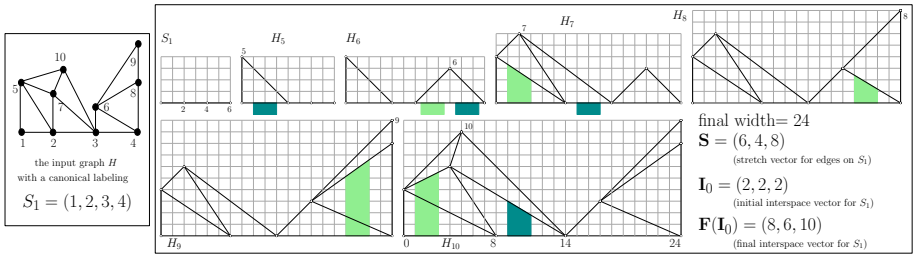


Figure 5.18: Example of execution of the shift algorithm on a half-4-scheme triangulation H with 10 vertices. We use $I_0 = (2, 2, 2)$ as initial interspace vector on the bottom path S_1.

H, H′ of G separated by P are H4ST (with S_2 as bottom path of the left component and S_4 as bottom-path of the right component), see Fig. 5.17(top). A straight-line drawing of a H4ST H is called *admissible* if S_1 is horizontal, S_2 and S_4 are vertical, and all edges in S_3 have slope in $\{-1, 0, +1\}$. By an extension of the shift paradigm underlying the FPP algorithm, Duncan et al. show that it is possible to obtain an admissible straight-line drawing of H on a grid of size $O(n \times n^2)$, more precisely a grid of size $O(n \times (d+1)n)$, where n is the number of vertices of H and d is the graph-distance between S_1 and S_3 (Fig. 5.18 illustrates the steps of this variation of the shift paradigm).

To obtain a straight-frame drawing of a 4ST G, decompose G along a river between S_1 and S_3 into two H4ST components H and H′, draw H and H′ using the shift algorithm (with S_2 as the bottom-path of H and S_4 as the bottom-path of H′), and then shift the left boundary of the component whose drawing has the smaller width so that the widths of the drawings of H and H′ coincide. Then rotate the drawing of H by $\pi/2$ clockwise, rotate the drawing of H′ by $\pi/2$ counterclockwise, and put the two drawing in front of each other, leaving enough horizontal space between them so that the edges connecting H to H′ have slope smaller than 1 in absolute value. Since the edges on the boundaries of (the rotated copies of) H and H′ are either vertical or of slope in $\{-1, +1\}$, the edges between H and H′ do not introduce crossings, so the resulting drawing of G is planar, see Fig. 5.17(bottom). Overall the grid-size is $O(n^2 \times n)$, more precisely $O(n(d+1) \times n)$, with d the graph-distance between S_2 and S_4.

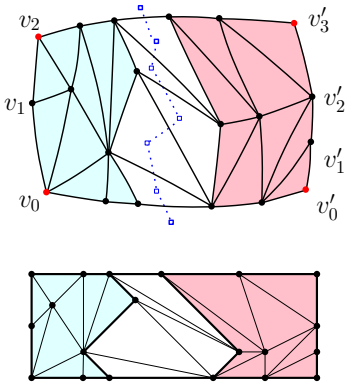


Figure 5.17: Illustration of the algorithm of Duncan et al. [DGK11] for drawing a 4ST G (top). The river between S_1 and S_3 decomposes G into two H4ST H, H′ (shaded). (bottom) The straight-frame drawing of G as given in [DGK11], which results from the two drawings of H, H′ put together.

5.4.2 *Our modified version of the algorithm: aligning vertical coordinates*

A first simple adaptation we do is to do all (abscissa) shift operations by 2 instead of doing them by 1. Given a H4ST G, let p be the number of edges of S_1. For a vector I of p even integers, I is called the *initial interspace vector*, consider the shift algorithm for G starting with S_1 drawn as a horizontal line with interspaces given by I. Let $F(I) = (f_1, \ldots, f_p)$ be the vector representing the interspaces between consecutive vertices on S_1 at the end of the algorithm; $F(I)$ is called the *final interspace vector*. For our purpose, the advantage of doing the shift operations by 2 is to guarantee that the components of $F(I)$ are even (see Fig. 5.18 for an example).

Lemma 5.6 *Let $F_0 = F(I_0)$ be the final interspace vector when using $I_0 = (2, 2, \ldots, 2)$ as initial interspace vector on S_1. Then for any vector F of p even integers such that $F \geqslant F_0$ (component-wise), it is possible to re-execute the shift-algorithm so as to have F as final interspace vector.*

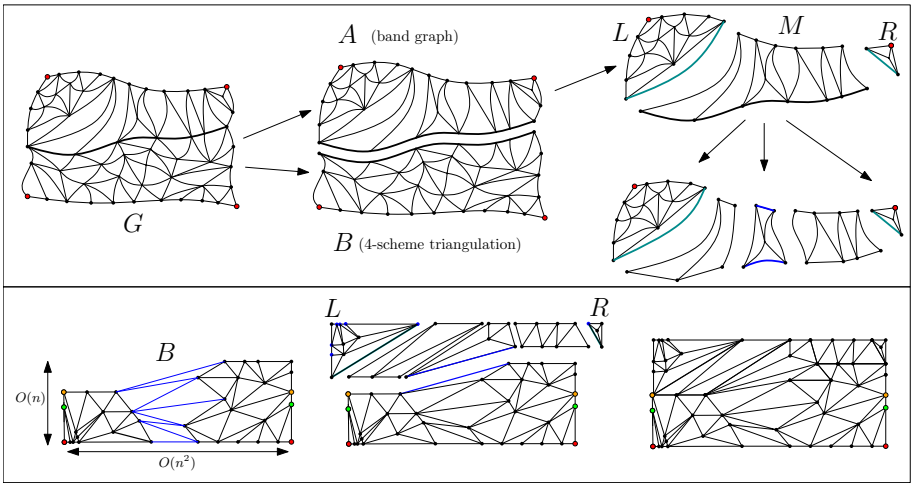


Figure 5.20: The top pictures illustrate our binary decomposition of a 4-scheme triangulation G (left), where we distinguish a chordless path P just below S_3 (shown bolder). Cutting along P yields two components: a 4-scheme triangulation B and a band-graph A which is further decomposed into 3 pieces L, M, R.

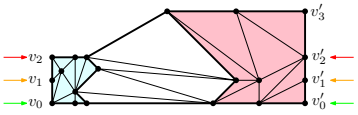


Figure 5.19: Our modified version of the algorithm of Duncan et al. [DGK11] (with an oblique edge on S_3). The application of Lemma 5.7 (for $m = 2$) ensures that the ordinates of the vertices on the left vertical path S_2 coincide with the ordinates of the corresponding vertices on the right path S_4. The input 4-scheme triangulation corresponds to the example given in Fig. 5.17.

ALIGNING VERTICES ON S_1 AND S_3. We can now give another strategy (suited in view of showing Theorem 5.5) for drawing a 4ST G, while guaranteeing that some of the vertices on the opposite sides of the frame will have the same vertical coordinates. To be more precise the resulting drawing is not straight-frame, but is straight-frame except for an oblique edge along S_3, and is such that, with $m = \min(|S_2|, |S_4|)$, the m first components of the interspace vectors along S_2 and S_4 (ordered from bottom to top) coincide (as illustrated in the example of Fig. 5.19).

At first, similarly as in [DGK11] we cut G along a river (between S_1 and S_3) into two components H and H′. Then we draw independently H and H′ using the shift algorithm (with width 2 strip insertions). Let F_0 (F_0') be the final interspace vector of H (resp. of H′), starting with initial interspace vector $(2, \ldots, 2)$. For $1 \leqslant i \leqslant m$ let u_i be the maximum of the ith components of F_0 and F_0'. Because of the property stated in Lemma 5.6, one can redraw H and H′ so that the m first components of the final interspace vectors of H and of H′ are $(u_1, \ldots, u_m)$. In addition one can check that the widths of both drawings are at most $8n$, and the two widths differ by at most $4n$. Similarly as in [DGK11] we rotate H (resp. H′) by $\pi/2$ clockwise (resp. counterclockwise) and place the drawings in front of each other, leaving horizontal space $8n$ between them, enough to draw the edges between H and H′ crossing-free. We obtain (see Fig. 5.19):

Lemma 5.7 *For any 4ST G with n vertices, and $m = \min(|S_2|, |S_4|)$, there is a straight-line drawing of G, where S_2 and S_4 are vertical with their interspace vectors equal at the m first components, S_1 is horizontal, and S_3 is horizontal except for an oblique edge of slope in $[-1/2, 1/2]$. The grid size is $O(n^2 \times n)$, more precisely $O(n(d+1) \times n)$ with d the graph-distance between S_2 and S_4, and the drawing can be computed in linear time.*

5.4.3 *A new binary decomposition for 4-scheme triangulations*

We now introduce a new way to decompose a 4ST G into two components A, B, in such a way that proving Theorem 5.5 will reduce to drawing B using Lemma 5.7, and then drawing A using a certain fixed outer frame. A path P is said to be *just below* S_3 if P connects a vertex of S_2 to a vertex of S_4, all non-extremal vertices of P avoid $S_2 \cup S_3 \cup S_4$, and each vertex on P (including the extremities) has at least one neighbour on S_3.

Lemma 5.8 *Each 4ST G has a chordless path just below S_3.*

Let P be a chordless path just below S_3. Cutting G along P yields two 4ST denoted A and B, with B below P and A above P. We draw B using Lemma 5.7. Then we have to draw A —which is called the *band-graph*— in such a way that the drawing obtained by pasting the drawing of B with the drawing of A yields a periodic drawing of G.

DRAWING THE BAND-GRAPH A (ON A GIVEN FIXED FRAME). To state the drawing result for A, we introduce the notion of *fixed frame*. Given a quasi-triangulation G, with C its outer cycle, a fixed frame κ for G is a crossing-free drawing of C on a regular grid $w \times h$ (w and h are the width and height of the fixed frame). For $\gamma \geqslant 1$ the grid is *refined by factor* γ by replacing each unit cell by a $\gamma \times \gamma$ regular grid. We say that the factor γ is *suitable* for the pair (G, κ) if, after refining the grid of κ by factor γ, G admits a straight-line drawing with κ as the outer face contour of the drawing, see Fig. 5.21 for an example.

Next Lemma states that, up to increase the vertex resolution of a linear factor, we can always draw a band-graph within a prescribed fixed frame (where the bottom path consists of horizontal segments except for an oblique edge, as in the top picture of Figure 5.22).

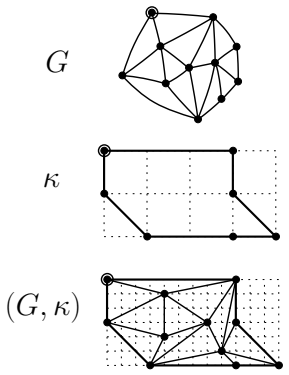


Figure 5.21: In order to draw a quasi-triangulation G in a fixed frame κ (on a grid of size 4×2), we use a refinement factor 3 for the grid (hence the factor 3 is suitable for (G, κ)).

Lemma 5.9 *For each $K \geqslant 1$ and any fixed frame κ for A (A is the band-graph of G, which has n vertices) of the form shown in the top drawing of Figure 5.22, there is γ_0 in $O(n \cdot \max(n, K))$ such that any even factor $\gamma \geqslant \gamma_0$ is suitable for (A, κ).*

The validity of this statement, for drawing the band-graph A of a 4ST G with n vertices obtained after cutting along a chordless path P, relies on a further chordal decomposition (see right pictures in Fig. 5.20).

Let a be the extremity of P on S_2 and b the extremity of P on S_4. Let u_a be the leftmost neighbour of a along S_3 and let u_b be the rightmost neighbour of b along S_3. Then cutting A along the edges $\{a, u_a\}$ and $\{b, u_b\}$ yields three pieces: a left-piece L, a middle piece M, and a right-piece R (as in Fig. 5.20). Note that L (and similarly R) is either empty (if u_a is the top-left corner of G) or otherwise is naturally a 3-scheme triangulation, a quasi-triangulation with three corners are a, u_a and the topleft corner of G: drawing L and R with a fixed frame requires a further decomposition along the chordal edges, if any, between the bottom and vertical paths as shown in the drawing of Fig. 5.23 (the details of the construction are omitted here and can be found in [CFK14]). Observe that M is naturally a 4ST such that $|S_2(M)| = |S_4(M)| = 1$ (so we can apply a variation of Lemma 5.7).

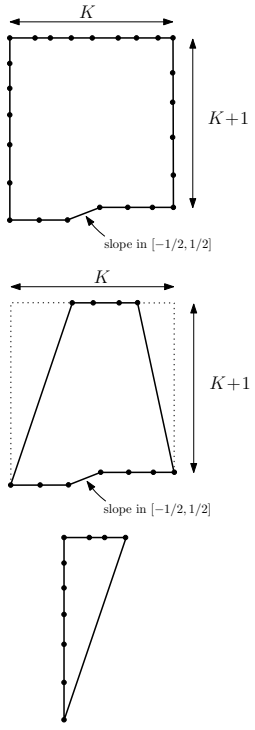


Figure 5.22: Left: frame for the band-graph A, middle: frame for the middle piece M of A, right: frame for the left piece L of A.

FINAL STEP: COMPUTING THE PERIODIC STRAIGHT-FRAME DRAWING. We claim that Lemma 5.7 together with Lemma 5.9 imply Theorem 5.5 (at the moment with grid-size $O(n^5 \times n^5)$). Indeed, once B is drawn using Lemma 5.7, one easily designs a fixed frame of the form of Fig. 5.22 and

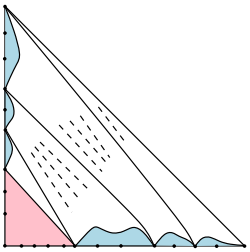


Figure 5.23: Chordal decomposition of a 3-scheme triangulation, with one side (hypotenuse) reduced to an edge.

such that pasting a drawing of A in this frame with the drawing of B yields a periodic drawing of G (note that the lower part of κ is determined by the property of fitting with the boundary-path $S_3(B)$, and the upper part of κ is determined by the property of fitting with the boundary-path $S_1(B)$ in order to get the periodicity property). Note that the parameter K equals the width of the drawing of B, which is $O(n^2)$. Since the refinement factor for A is in $O(n \cdot \max(n, K))$, the grid-size of the resulting drawing of G is $O(K^2 n \times K^2 n)$, which is $O(n^5 \times n^5)$.

5.4.4 *Final remarks*

DEALING WITH CHORDS BETWEEN BOUNDARY PATHS. In order to carry out the decomposition of the input 4ST G based on the path just below S_3 we have assumed that there is no chord between S_1 and S_3. If there is a chord between S_1 and S_3 but there is no chord between S_2 and S_4, then we can just rotate G by $\pi/2$ (so as to exchange S_1, S_3 with S_2, S_4). If there is also a chord between S_2 and S_4 then it must be incident to a corner vertex: this case requires a slight modification of our construction (a more detailed discussion is provided in [CFK14]).

IMPROVING THE GRID SIZE. We would like to point out that in the case where there is no chord incident to a corner of G, then G actually admits a periodic straight-frame drawing of grid-size $O(n^4 \times n^4)$. This fact relies mainly on the following remark. Let us assume that the input 4-scheme triangulation G has n vertices and there are no chords between boundary paths (let d_h be the graph-distance between S_1 and S_3, and let d_v be the graph-distance between S_2 and S_4). Then it follows from the Menger vertex-disjoint theorem and the fact that G is innerly triangulated that $d_h \times d_v \leqslant n$, hence $\min(d_h, d_v) \leqslant n^{1/2}$: this implies that, up to possibly rotating G by $\pi/2$ to ensure that $d_h \leqslant d_v$, the Duncan et al. algorithm gives a grid-size $O(n \times n^{3/2})$ (instead of the worst case $O(n \times n^2)$ grid-size mentioned in Section 5.4.3). A few more technical arguments (omitted here) should be required to deal with case where there are chords between boundary paths.

5.5 SPHERICAL DRAWINGS OF PLANAR GRAPHS

The notion of planar drawings can be naturally generalized to the spherical case: the main difference is that edges are mapped to *geodesic arcs* on the unit sphere $\mathbb{S}^2$, which are minor arcs of great circles (obtained as intersection of $\mathbb{S}^2$ with a hyperplane passing through the origin). A *geodesic drawing* of a map should preserve the cyclic order of neighbors around each vertex (such an embedding is unique for triangulations, up to reflexions of the sphere). As in the planar case, we would aim to obtain *crossing-free* geodesic drawings, where geodesic arcs do not intersect (except at their extremities): these are referred to as *spherical drawings*. Sometimes, the weaker notion of *spherical parameterization* (an homeomorphism between an input mesh and $\mathbb{S}^2$) is considered for dealing with applications in the geometry processing domain (e.g. for dealing with mesh morphing): while the bijectivity between the mesh and $\mathbb{S}^2$ is guaranteed, there are no guarantees that the triangle faces are mapped to spherical triangles with no overlaps (obviously a spherical drawing leads to a spherical parameterization).

5.5.1 *Related works*

Spherical parameterization via Tutte barycentric method (in the plane)

A simple solution to the spherical parameterization problem consists in planarizing the input graph and to apply the standard Tutte barycentric method (this can be done in several ways, as illustrated below): one main advantage is that one can preserve some of the aesthetic appeal of Tutte's planar drawings. The main drawback is that these maps are bijective but cannot produce in general a crossing-free spherical drawing (straight-line segments in the plane are not mapped to geodesics by inverse stereographic or polar-to-cartesian projections). Nevertheless, the methods based on planarization are extremely simple to implement (and rather efficient in practice) and can be used as *initial placers* for more sophisticated iterative spherical layout algorithms (an experimental evaluation will be provided in Section 5.6).

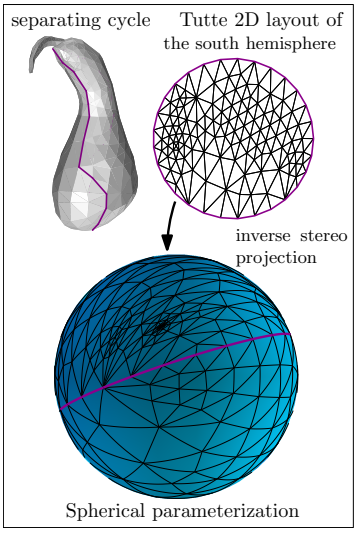


Figure 5.24: A spherical parameterization of the `gourd` graph obtained combining a Tutte's planar parameterization with an inverse stereo projection.

INVERSE STEREO PROJECTION LAYOUT (`isp`). The solution suggested by Saba et al. [Sab+05] (see Fig. 5.24). consists in partitioning the input graph G into two components homeomorphic to a disk: this is achieved by computing a small simple cycle separator (having $O(\sqrt{n})$ vertices) whose removal produces a balanced partition (G^S, G^N) of the faces of G. The two graphs G^S and G^N are then drawn in the plane using Tutte's barycentric method: boundary vertices lying on the separator are mapped on the unit disk. Combining a Moebius inversion with the inverse of a stereographic projection we obtain a spherical parameterization of the input graph.

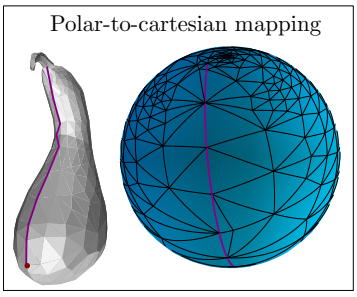


Figure 5.25: A spherical parameterization of the `gourd` graph obtained via Tutte's planar parameterization.

POLAR-TO-CARTESIAN LAYOUT (`pc`). The approach adopted in Zayer, Rössl, and Seidel [ZRS06] relies on a different planarization step (the result is shown in Fig. 5.25): the edges along a simple path from a south pole v^S to a north pole v^N. A planar rectangular layout is computed by applying Tutte parameterization with respect to the azimuthal angle $\theta \in (0, 2\pi)$ and to the polar angle $\phi \in [0, \pi]$.

Tutte barycentric embedding and force-directed layouts on the sphere

ITERATIVE RELAXATION: PROJECTED GAUSS-SEIDEL. The first method can be viewed as an adaptation of the iterative scheme solving Tutte equations. This scheme consists in moving points on the sphere in tangential direction in order to minimize the spring energy (its pseudo-code is shown in Fig. 5.26)

$$\mathcal{E} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j\in N(i)} w_{ij}\|\mathbf{x}(v_i)-\mathbf{x}(v_j)\|^2 \tag{5.1}$$

with the only constraint $\|\mathbf{x}(v_i)\| = 1$ for $i = 1 \dots n$ (in our implementations we use uniform weights w_{ij}, as in Tutte's work). As opposed to the planar case, there are no boundary constraints on the sphere, which makes the resulting layouts collapse in many cases to degenerate solutions. As observed in [GGS03; Sab+05] this method does not always converge to a valid spherical drawing, and its practical performance strongly depends on the geometry of the starting initial layout $\mathbf{x}^0$. While not having theoretical guarantees, this method is quite fast allowing to quickly decrease the residual error: it thus can be used in a first phase and combined with more stable iterative schemes leading in practice to better convergence results [Sab+05] (still lacking of rigorous theoretical guarantees).

Projected Gauss-Seidel($\mathbf{x}^0$, λ, ε)
$r = 0$; // *iteration counter*
do {
  for($i = 1$; $i \le n$; i++) {
    $\mathbf{s} = (1-\lambda)\mathbf{x}^r(v_i) + \lambda\sum_j w_{ij}\mathbf{x}^r(v_j)$
    $\mathbf{x}^{r+1}(v_i) = \frac{\mathbf{s}}{\|\mathbf{s}\|}$
  }
  r++;
} while ($\|\mathbf{x}^r - \mathbf{x}^{r-1}\| > \varepsilon$))

Figure 5.26: The pseudo-code of the *Projected Gauss-Seidel* method.

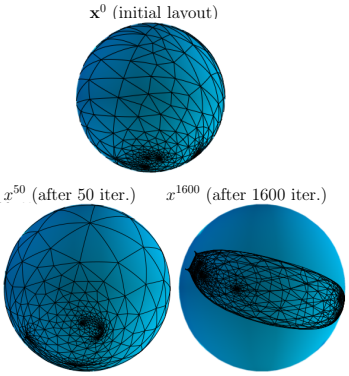


Figure 5.27: Using iterative solvers the drawing can sometimes collapses to degenerate layouts.

ALEXA'S METHOD. In order to avoid the collapse of the layout, without using artificial constraints, Alexa [Ale00] modified the iterative relaxation above by penalizing long edges (tending to move vertices in a same hemisphere). More precisely, the vertex v_i is moved according to a displacement $\triangle_i = c\,\frac{1}{deg(v_i)}\sum_j(\mathbf{x}(v_i)-\mathbf{x}(v_j))\|\mathbf{x}(v_i)-\mathbf{x}(v_j)\|$ and then reprojected on the sphere. The parameter c regulates the step length, and can be chosen to be proportional to the inverse of the longest edge incident to a vertex, improving the convergence speed.

FORCE-DIRECTED LAYOUTS ON THE SPHERE. While spring embedders are originally designed to produce 2D or 3D layouts, one can adapt them to spherical geometry. We have implemented the standard *spring-electrical model* introduced in [FR91] (referred to as FR), and its spherical version [5] following the framework described by Kobourov and Wampler [KW05] (called Spherical FR). As in [FR91] we compute attractive forces (between adjacent vertices) and repulsive forces (for any pair of vertices) acting on vertex u, defined by:

$$F_a(u) = \sum_{(u,v)\in E}\frac{\|\mathbf{x}(u)-\mathbf{x}(v)\|}{K}(\mathbf{x}(u)-\mathbf{x}(v)),\ F_r(u) = \sum_{v\in V, v\neq u}\frac{-CK^2(\mathbf{x}(v)-\mathbf{x}(u))}{\|\mathbf{x}(u)-\mathbf{x}(v)\|^2}$$

where the values C (the strength of the forces) and K (the optimal distance) are scale parameters. When implementing the spherical version, we must pay attention to some details: for instance we shift the repulsive forces by a constant term, making the force acting on pairs of antipodal vertices zero.

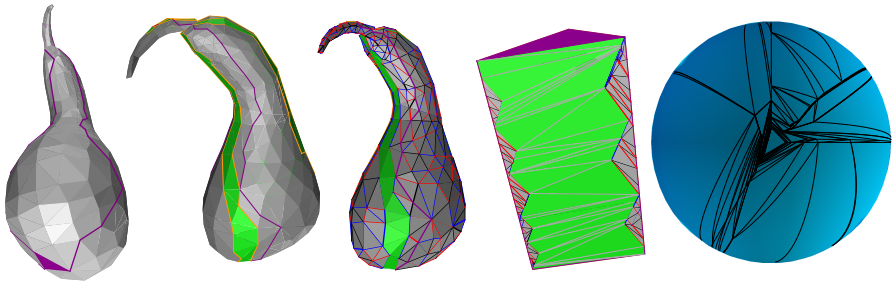


Figure 5.28: Computation of a spherical drawing based on a prism layout of the `gourd` graph (326 vertices). Three vertex-disjoint chord-free paths lead to the partition of the faces of G into three narrow 4-scheme triangulations G_0^C, G_1^C and G_2^C which are each separated by one river (green faces). Once the three planar layouts are computed (refer to Fig. 5.29) we can glue together the identified sides and obtain a 3D prism $\mathcal{M}$: its central projection on the sphere produces a spherical drawing.

5.6 OUR CONTRIBUTION: A FAST SPHERICAL DRAWING WITH THEORETICAL GUARANTEES

We now provide an overview of our algorithm for computing a spherical drawing of a planar triangulation $\mathcal{T}$ in linear time, called `SFPP` layout (see Fig. 5.28 for an illustration). Our solution relies on the results sketched in Section 5.4: more precisely, we apply our two-pass implementation of the shift paradigm, described in Section 5.4.2, which allows us obtain a straight-frame drawings with vertices aligned on opposite sides. A more detailed presentation can be found in [CDF18a].

STEP 1: MESH SEGMENTATION. Assuming that there are two non-adjacent faces f^N and f^S (with no common incident vertices [6]), one can compute in linear time [7] 3 disjoint and chord-free paths P_0, P_1 and P_2 from f^S to f^N. Denote by u_0^N, u_1^N and u_2^N the three vertices of f^N on P_0, P_1 and P_2 (define similarly the three neighbors u_0^S, u_1^S, u_2^S of the face f^S). We first compute a partition of the faces of $\mathcal{T}$ into 3 4-scheme triangulations, denoted by G_0^C, G_1^C and G_2^C, cutting $\mathcal{T}$ along the paths above and removing f^S and f^N. The first pair of opposite sides only consist of an edge (drawn as vertical segment in Fig. 5.29), while the remaining pair of opposite sides contains vertices lying on P_i and P_{i+1} respectively (indices being modulo 3): according to these definitions, G_i^C and G_{i+1}^C share the vertices lying on P_{i+1} (drawn as a path of horizontal segments in Fig. 5.29).

Note that $\mathcal{T}$ can be seen as a *prism* (as illustrated in Fig. 5.28), with f^N and f^S as the two triangular faces and with G_0^C, G_1^C and G_2^C occupying the three lateral quadrangular faces of the prism.

5 The experimental evaluation of our implementation of `Spherical FR` method is given in Section 5.7

6 If this is not the case, one can force the existence of two such faces by adding a new triangle t within a face (and adding edges so as to triangulate the area between t and the face-contour)

7 Once again Schnyder woods provide a nice and efficient way for computing paths P_i. Taking f^N as the outer face, where v_0, v_1, v_2 are the outer vertices in clockwise (CW) order and inserting a vertex v of degree 3 inside f^S, we compute a Schnyder wood of the resulting triangulation. Let P_0, P_1, P_2 be the directed paths in respective colors $0, 1, 2$ starting from v: the well-known properties of Schnyder woods ensure that these paths are chord-free and disjoint except at v, ending at the 3 vertices v_0, v_1, v_2 of f. Deleting v and its 3 incident edges, (and thus deleting the starting edge in each of P_0, P_1, P_2) we obtain a triple of disjoint chord-free paths from f^S to f^N. Let u_0, u_1, u_2 be the vertices on f^S incident to P_0, P_1, P_2.

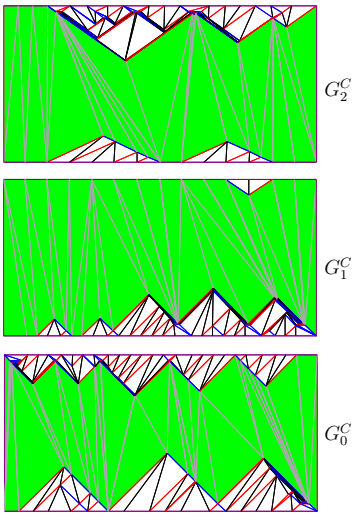


Figure 5.29: Our variant of the FPP algorithm allows to produce straight-frame layouts of the three 4-scheme triangulations G_0^C, G_1^C and G_2^C, where boundary vertex locations do match on identified horizontal sides (corresponding to the common path P_{i+1}) shared by G_i^C and G_{i+1}^C).

STEP 2: COMPATIBLE DRAWINGS OF RECTANGULAR FRAMES. We apply the strategy sketched in Section 5.4 to obtain three rectangular layouts of G_0^C, G_1^C and G_2^C (as illustrated in Fig. 5.29): this algorithm first separates each G_i^C into two sub-graphs by removing the so-called river (an outerplanar graph consisting of a face-connected set of triangles which corresponds to a simple path in the dual graph, starting at f^S and going toward f^N). The two-subgraphs of G_i^C (bottom and top white regions in Fig. 5.29) are then processed making use of the *canonical labeling* defined in [DGK11]: the resulting layouts are stretched in order to fit into a rectangular frame (observe that after this first pass the boundary vertices corresponding to the path P_i are not aligned). Just observe that in our case a pair of opposite sides only consists of two edges (vertical edges in Fig. 5.29), which leads to an algorithm considerably simpler to implement in practice. Finally, before re-inserting the set of edges in the river, we apply a two-phases adaptation of the shift algorithm similar to the one described in Section 5.4.2 to obtain a planar grid drawing of each 4-scheme triangulation G_i^C, such that the positions of vertices on the path P_{i+1} in G_i^C do match the positions of corresponding vertices on P_{i+1} in G_{i+1}^C.

STEP 3: SPHERICAL LAYOUT. To conclude, we glue together the drawings of G_0^C, G_1^C and G_2^C computed above in order to obtain a drawing of $\mathcal{T}$ on a triangular prism. By a translation within the 3D ambient space we can make the origin coincides with the center of mass of the prism (upon seeing it as a solid polyhedron). Then a central projection from the origin maps each vertex on the prism to a point on the sphere: each edge (u, v) is mapped to a geodesic arc, obtained by intersecting the sphere with the plane passing trough the origin and the segment relying u and v on the prism (crossings are forbidden since the map is bijective).

Note that the prism has its 3 lateral faces G_i^C of area $O(n) \times O(n)$: this is implied by the resolution achieved by the algorithm of Duncan et. al [DGK11] and on the fact that the two opposite sides $(u_i^N, \ldots, u_i^S)$ and $(u_{i+1}^N, \ldots, u_{i+1}^S)$ of G_i^C are at distance 1.

Theorem 5.10 ([CDF18a]) *Let $\mathcal{T}$ be a planar triangulation of size n, having two non-adjacent faces f and f'. Then one can compute in $O(n)$ time a spherical drawing of $\mathcal{T}$, where edges are drawn as (non-crossing) geodesic arcs of length $\Omega(\frac{1}{n})$.*

It is worth noting that the algorithmic tools involved in the theorem above are reasonable simple to combine, leading to an extremely fast implementation (see Section 5.7).

5.7 EXPERIMENTAL RESULTS

5.7.1 *Spherical drawings*

Here we provide an experimental evaluation of the combinatorial tools developed in the previous section for computing spherical drawings [8].

While the resulting SFPP layouts are rather unpleasing (as for the planar FPP algorithm), our solution is able to compute spherical drawings that are

8 The experimental results of this section are taken from [CDF18a].

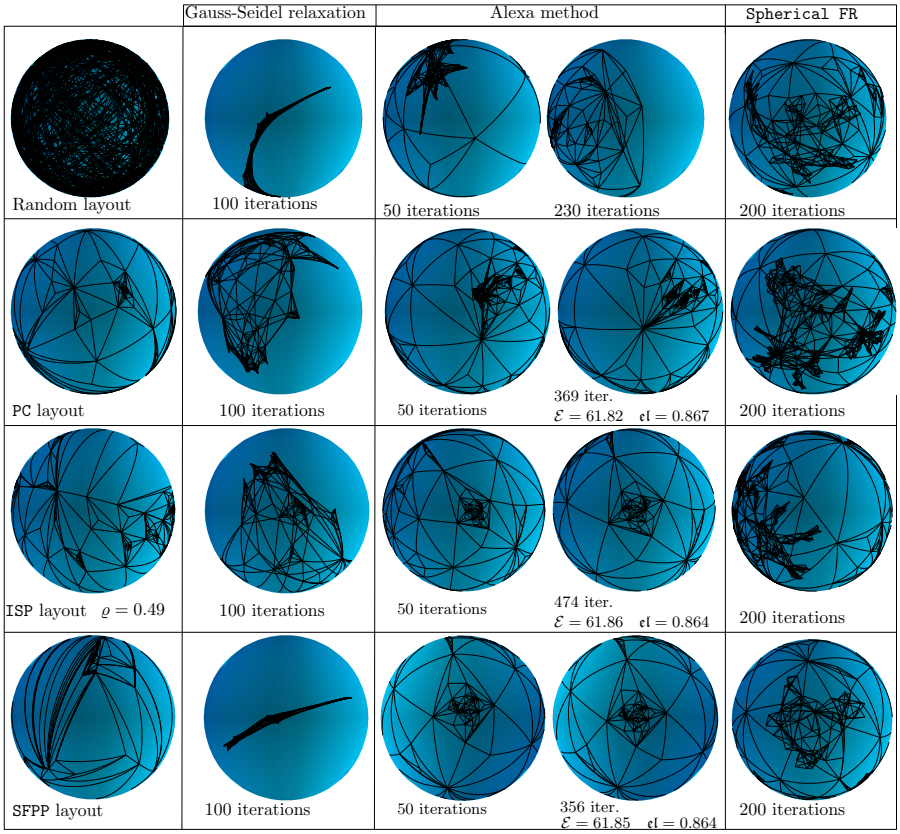


Figure 5.30: Spherical layouts of a random triangulation with 1K faces. While the projected Gauss-Seidel relaxation always collapses, Alexa method is more robust, but also fails when starting from a random initial layout. When using the ISP, PC or our SFPP layouts Alexa method converges toward a crossing-free layout: starting from the SFPP layout allows getting the same aesthetic criteria as the ISP or the PC layouts (with even less iterations). Spring embedders [FK12] (Spherical FR) prevent from reaching a degenerate configuration, but have some difficulties to unfold the layout.

good starting points for both iterative methods and spring embedders: our intuition is that an initial layout which is crossing-free could be of some help to unfold the graph in order to obtain nicer spherical drawings, in a more efficient and robust way, as confirmed empirically by the experiments presented below.

COMPARISON OF RUNNING TIMES. The plots of Fig. 5.31 clearly show that our SFPP algorithm has an asymptotic linear-time behavior and in practice is much faster than the ISP and PC methods based on the resolution of linear systems, for which we use the conjugate solver of the MTJ library. Observe that a slightly better performance can be achieved with more sophisticated schemes or tools (e.g. Matlab solvers) as done in [AL15; Sab+05]. Nevertheless the timing cost still remains much larger than ours: as reported in [AL15] the parameterization of the Dragon graph (655k vertices) requires 19 seconds for solving the linear systems (on a 3.5GHz Intel i7 CPU), while the computation of our SFPP layout requires less than 1.2 seconds.

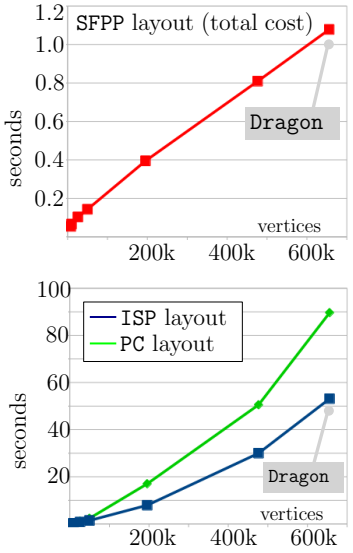


Figure 5.31: (top) Runtime costs for the computation of our SFPP layouts. (bottom) Timing cost for solving the linear systems for the ISP and PC layouts (we use the MTJ conjugate gradient solver with a numeric tolerance of 10^{-6}). All results are expressed in seconds.

QUANTITATIVE EVALUATION OF AESTHETIC CRITERIA. In order to obtain a quantitative evaluation of the layout quality we compute the spring

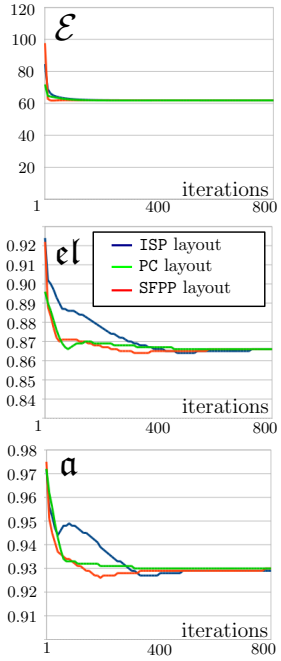


Figure 5.32: Comparison of spherical layouts of a random triangulation with 1K faces: we evaluate the performances of Alexa method, starting from several initial layouts. The charts above show the plot of the energy, edge lengths and areas statistics computed when running 800 iterations of Alexa method (we compute these statistics every 10 iterations).

energy $\mathcal{E}$ defined by Eq. 5.1 and two metrics measuring the edge lengths and the triangle areas (refer to the plots of Fig. 5.32). As suggested in [FK12] we compute the average percent deviation of edge lengths, according to

$$\mathfrak{el} := 1 - \left(\frac{1}{|E|} \sum_{e \in E} \frac{|l_g(e) - l_{avg}|}{\max(l_{avg}, l_{max} - l_{avg})} \right) \tag{5.2}$$

where $l_g(e)$ denotes the geodesic distance of the edge e, and l_{avg} (resp. l_{max}) is the average geodesic edge length (resp. maximal geodesic edge length) in the layout. Similarly we compute the average percent deviation of triangle areas, denoted by $\mathfrak{a}$. The metrics $\mathfrak{el}$ and $\mathfrak{a}$ take values in $[0 \dots 1]$, and higher values indicate more uniform edge lengths and triangle areas [9].

Comparisons of layouts

We evaluate and compare the spherical layouts obtained applying iterative schemes (Alexa's method, projected Gauss-Seidel scheme and the `Spherical FR` force-directed embedder), when starting from distinct initial layouts (initial random locations, `ISP`, `PC` and our `SFPP`).

All our tests confirm that starting with random vertex locations is almost always a bad choice, since iterative methods lead in most cases to a collapse before reaching a valid spherical drawing (spherical spring embedders do not have this problem, but cannot always eliminate edge crossings, see Fig. 5.30). Our experiments (see Fig. 5.30) also confirm a well known fact: Alexa's method is more robust compared to the projected Gauss-Seidel relaxation, leading almost always to a valid configuration without collapsing (more tests and statistics, also including real-world 3D meshes, can be found in [CDF18a]).

LAYOUT OF RANDOM TRIANGULATIONS. When drawing random triangulations the performances obtained starting from our `SFPP` layout are often better than the ones achieved using the `ISP` layout (and similar to the ones of the `PC` layout). As illustrated by the pictures in Fig. 5.30 and 5.34, Alexa's method is able to reach a non-crossing configuration requiring less iterations when using our `SFPP` layout instead of `ISP` layout: this is observed in most of our experiments, and clearly confirmed by the plots of the energy and statistics $\mathfrak{el}$ and $\mathfrak{a}$ that converge faster to the values of the final layout (see charts in Fig. 5.32).

5.7.2 *Spherical preprocessing for Euclidean spring embedders*

We investigate the use of spherical drawings as initial placers for spring embedders in 3D space. The fact of recovering the original topological shape of the graph, at least in the case of graphs that have a clear underlying geometric structure, is an important and well known ability of spring embedders. This occurs for the case of regular graphs used in Geometry Processing (the pictures in Fig. 5.33 show a few force-directed layouts of the `cow` graph), but also for the case of random maps, for which spring embedding algorithms (applied in the 3D ambient space) yield graph layouts that greatly capture the topological and structural features of the map (the genus of the surface is visible, the "central charge" of the model is reflected by the presence of

9 Observe that one common metric considered in the geometric processing community is the (angle) distortion: in our case this metric cannot be taken into account since our input is a combinatorial structure (without any geometric embedding).

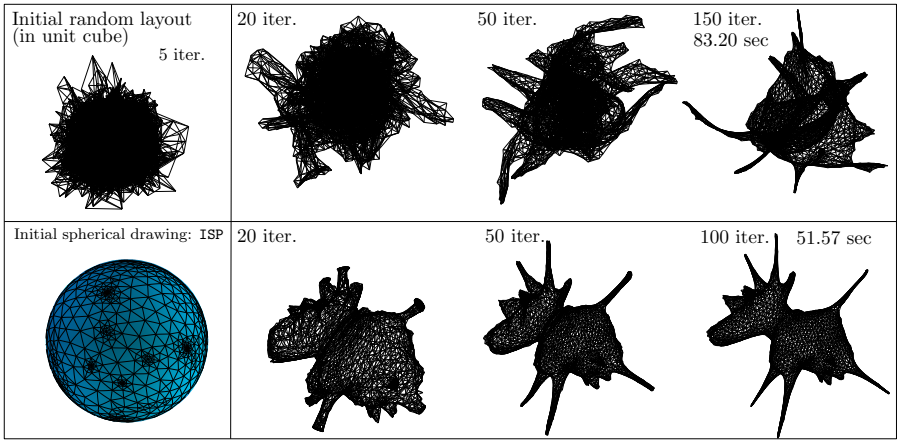


Figure 5.33: Use of spherical drawings as initial placers for force-directed methods: we compute the layouts of the cow graph (2904 vertices, 5804 faces) using our 3D implementation of the FR spring embedder [FR91]. Starting from a good initial shape helps to untangle faster the graph, leading very often to better local minima. A quantitative confirmation is provided by the plots in Fig. 5.35.

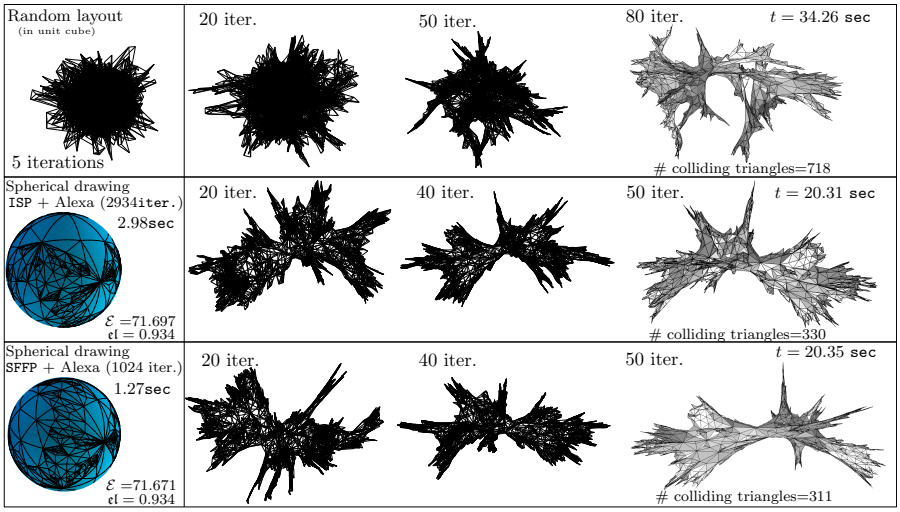


Figure 5.34: Force-directed drawings of a random planar triangulation with 5K faces: spherical drawings are used as initial placers for the 3D version of the FR spring embedder [FR91].

spikes, etc.) [10]. While common software and libraries for graph visualization (e.g. GraphViz [Ell+01], Gephi [BHJ09], GraphStream) provide implementations of many force-directed models, as far as we know they never try to exploit the strong combinatorial structure of surface-like graphs.

DISCUSSION OF EXPERIMENTAL RESULTS. Our main goal is to show empirically that starting from a nice initial shape that captures the topological structure of the input graph greatly improves the convergence speed and layout quality. In our experiments (see Figures 5.33 and 5.34) we run

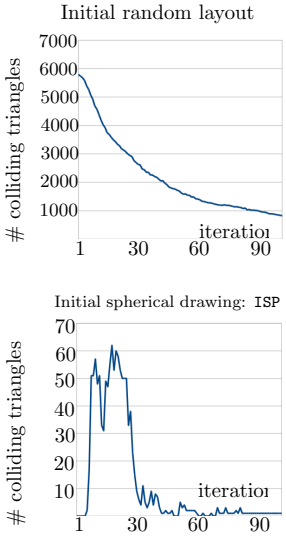


Figure 5.35: 3D FR layout of the cow graph (2904 vertices) obtained starting from the two different initial placements drawn in Fig. 5.33. We plot the number of colliding 3D triangles, over 100 iterations of the FR

10 A great variety of such representations can be seen at the very nice simulation gallery of Jérémie Bettinelli (http://www.normalesup.org/~bettinel/simul.html).

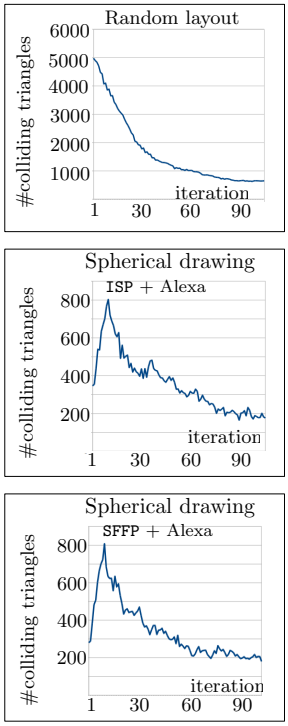


Figure 5.36: 3D layout of a random planar triangulation with 5K faces: spherical drawings are used as initial placers for the 3D version of the FR spring embedder [FR91]. We plot the number of colliding 3D triangles, over 100 iterations of the FR algorithm.

our 3D implementation of the spring electrical model FR [FR91] [11] starting from several distinct initial layouts.

In order to quantify the layout quality, we evaluate the number of self-intersections of the resulting 3D shape during the iterative computation process [12]. To be more precise, we plot over the first 100 iterations the number of triangle faces that have a collision with a non adjacent triangle in 3D space (see charts of Fig. 5.35 and 5.36).

Our experiments clearly confirm the visual intuition suggested by pictures: when starting from a good initial shape the force-directed layouts seem to evolve according to an inflating process, which leads to better and faster untangle the graph layout. This phenomenon is observed in all our tests (on several mesh-like graphs and synthetic data): experimental evidence shows that an initial spherical drawing is a good starting point helping the spring embedder to reach nicer layout aesthetics and also to improve the runtime performances. Finally observe that from the computational point of view the computation of a spherical drawing has a negligible cost: iterative schemes (e.g. Alexa method) require $O(n)$ time per iteration, which must be compared to the complexity cost of force-directed methods, requiring between $O(n^2)$ or $O(n \log n)$ time per iteration (depending on the repulsive force calculation scheme).

11 In our implementation of the FR algorithm we make use of exact force computation and we adopt the cooling system proposed in [Walo3] (with repulsive strength $C = 0.1$).

12 We compute the intersections between all pairs of non adjacent triangles running a brute-force algorithm: the runtimes reported in Fig. 5.33 and 5.34 do not count the cost of computing the triangle collisions.

6

SOME EXPERIMENTAL RESULTS ON SCHNYDER WOODS

In this section we provide some experimental results [1] involving both planar and higher genus Schnyder woods: in particular we wrote a Java program that performs an exhaustive generation of all possible 3-orientations and Schnyder woods for surfaces of low genus. In the case of tiny triangulations (having at most a few tens of vertices) our generator is fast enough to list all distinct Schnyder woods in a reasonable amount of time (up to a few hours): this allows us to compute some statistics and test some conjectures.

6.1 EXPERIMENTAL RESULTS ON HIGHER GENUS SCHNYDER WOODS

The results presented in this section are obtained running our Schnyder wood generator on all simple triangulations of genus 1 and 2. All distinct triangulations can be generated using the `surftri` tool developed by Thom Sulanke [Sul17; SL09] which produces the collection of all triangulations of a surface (up to a given size) starting from the set of *irreducible triangulations* of that surface. Table 6.1 reports the enumeration of triangulations of the torus and the double torus for $n \leqslant 11$.

n	# irreducible triangulations	#triangulations ($g = 1$)
7	1	1
8	4	7
9	15	112
10	1	2109
11	–	37867

n	# irreducible triangulations	#triangulations ($g = 2$)
7	–	–
8	–	–
9	–	–
10	865	865
11	26276	113506

Figure 6.1: Number of simple triangulations of genus 1 and 2 (results are obtained using `surftri`).

A triangulation is irreducible *if there is no contractible edge.*

MONOCHROMATIC CYCLES IN TOROIDAL SCHNYDER WOODS. For each tiny toroidal triangulation, we have computed all distinct 3-orientations corresponding to toroidal Schnyder woods that satisfy the local condition (T1): then we have checked whether there exists a Schnyder wood for which the edges of the same color define a connected graph (having thus 3 monochromatic cycles, one for each color). According to our experiments, this is true for all simple toroidal triangulations with at most 11 vertices, providing a partial empirical confirmation to an open question stated in [GL14].

SCHNYDER WOODS FOR THE DOUBLE TORUS. It is known that any simple triangulation of genus $g \geqslant 1$ admits an edge orientation such that for every vertex the outdegree is divisible by 3 and at least 3 (this result, proved by Albar, Gonçalves, and Knauer [AGK16], confirms a conjecture of Barát and Thomassen [BT06]). As already mentioned, already for $g = 1$ there are 3-orientations that do not correspond to Schnyder woods (recall the example of Fig. 3.1): but Theorem 3.10 ensures that any toroidal triangulation admits a Schnyder wood (so, there is at least one 3-orientation whose corresponding edge coloring satisfies the (T1) condition of Definition 3.9). For $g \geqslant 2$ a similar question naturally arises: one could ask whether it would be always possible to find a 3-orientation as defined above (the outdegree is a multiple of 3 and there are no sinks) that yields an edge coloring for which every vertex satisfies a *multiple* local Schnyder condition, as depicted in Fig. 6.2. This question was open until very recently: as shown in its PhD dissertation by Jason Suagee [Sua21], we now know that any simple triangulation of genus $g \geqslant 1$ having edge-width at least $40(2^g - 1)$ admits such a kind of Schnyder wood.

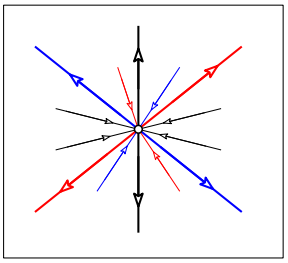


Figure 6.2: Multiple local Schnyder condition: the outdegree of every vertex is a positive multiple of 3.

1 Many experimental results presented in this chapter (Sections 6.2, 6.3 and 6.4) are taken from [Cas19].

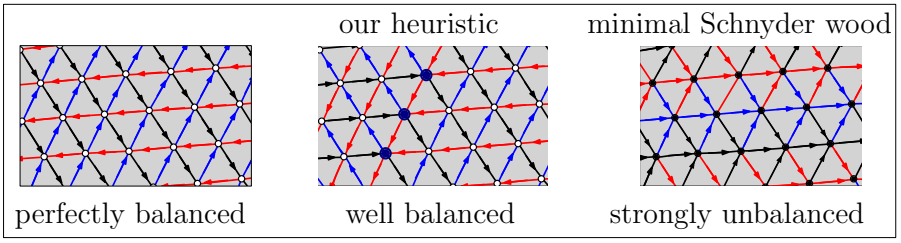


Figure 6.3: Three Schnyder woods of the same portion of a spherical grid (the whole mesh is shown in Fig. 6.4): our heuristic leads to a majority of balanced vertices (white circles), while the minimal Schnyder wood is strongly unbalanced.

Our experiments gives an empirical confirmation of this result and suggest that the assumption on the edge-width is maybe unnecessary. We generated all possible 3-orientations (defined as above) for all distinct simple triangulations of genus 2 with at most 11 vertices, and checked the local Schnyder condition at every vertex. As we observed, for any such triangulation there exists a generalized Schnyder wood satisfying the multiple Schnyder condition of Fig. 6.2, with no restriction on the edge-width.

Unfortunately, we cannot check the existence of Schnyder woods for triangulations embedded on surfaces of genus $g \geqslant 3$: in order to run the `surftri` tool one needs the list of all irreducible triangulations of genus 3 and such a list is not provided with the `surftri` tool.

6.2 BALANCE OF PLANAR SCHNYDER WOODS

MOTIVATION. The main idea motivating the experiments of this section is that, in practice, most real-word graphs exhibit strong regularities which make them far from the random and pathological cases. Based on this remark, we want to explore the possibility to exploit this regularity in order to obtain better results for algorithmic problems involving Schnyder woods. As far as we know, the problem of providing an adaptive analysis of Schnyder woods taking into account the graph regularity has not been investigated so far: in particular our goal is to evaluate the role of *balanced* Schnyder woods (where, intuitively, the number of incoming edges of each color at inner vertices is more or less the same). Here we provide empirical evidence about the fact that balanced Schnyder woods can lead to fast solutions achieving good results in practice, especially for real-world graphs: as test applications, we evaluate the layout quality of a Schnyder drawings (Section 6.3) and we consider the problem of computing small separators for planar graphs (Section 6.4).

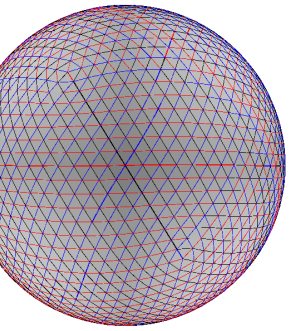


Figure 6.4: A spherical triangulation endowed with a well balanced Schnyder wood.

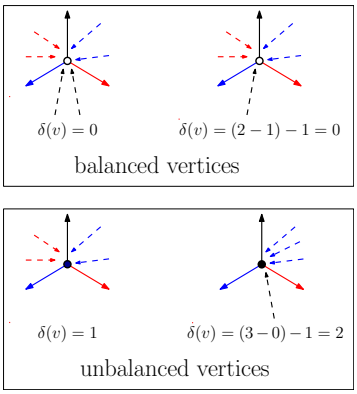


Figure 6.5: Balanced vertices have defect $\delta = 0$.

MEASURING THE BALANCE. The first step is to provide some notion of measure of the balance of a Schnyder wood: given an inner vertex v of degree $\deg(v)$ having $\mathrm{indeg}_i(v)$ incoming edges of color i (for $i \in \{0,1,2\}$), we define its *defect* as $\delta(v) = \max_i \mathrm{indeg}_i(v) - \min_i \mathrm{indeg}_i(v)$ if $\deg(v)$ is a multiple of 3, and $\delta(v) = \max_i \mathrm{indeg}_i(v) - \min_i \mathrm{indeg}_i(v) - 1$ otherwise (refer to Fig. 6.5 for an illustration). We say that a vertex is *balanced* if $\delta(v) = 0$ and a Schnyder wood is *well balanced* if a majority of vertices have a small defect. Another interesting parameter, closely related to the defect defined above, is the number of 3-colored faces [2]. For regular graphs is possible, in principle, to get a Schnyder wood that is perfectly balanced: $\delta(v) = 0$ almost everywhere and most faces are 3-colored, as shown in Fig. 6.3(left). In practice many Schnyder woods are unbalanced and we are not aware of existing theoretical or empirical results on the balance of Schnyder woods.

2 Observe that there are a number of interesting results on Schnyder woods involving the number of 3-colored faces (refer to [Bono2] for more details).

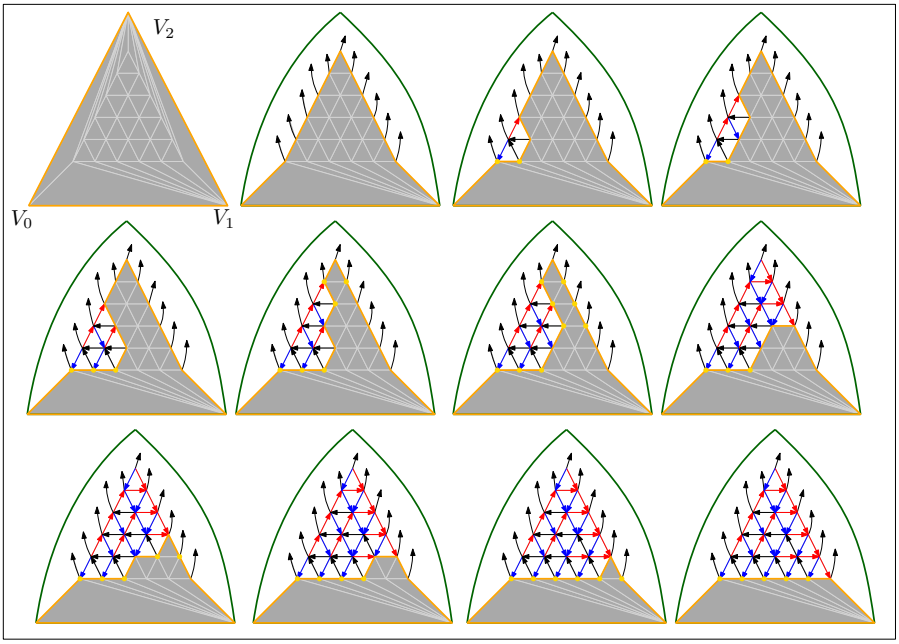


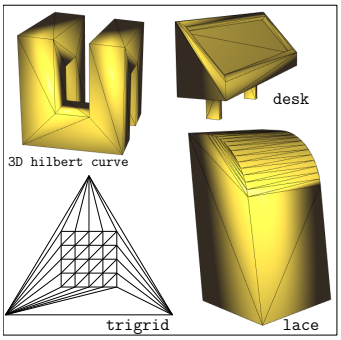
Figure 6.6: Example of execution of our heuristic algorithm for the computation of a well balanced Schnyder wood. The pseudo-code is given in Fig. 6.7.

6.2.1 *Our heuristic for well balanced Schnyder woods*

Observe that the vertex shelling procedure for computing Schnyder woods and canonical ordering (Section 2.2) has many degrees of freedom: at each step the choice of the vertex to be removed can possibly lead to a different Schnyder wood. In order to get as much as possible balanced vertices, we retard the removal of some vertices according to a balance priority (an example of execution of our heuristic is illustrated in Fig. 6.6). At a given step of the shelling procedure, we define the *priority* of a vertex v as the total number incoming incident edges that already have been assigned a color and orientation (toward v).

Then we make use of very simple (truncated) priority rule: among the free vertices on the outer cycle B (the boundary of the outer face) we select one with maximal priority to be removed. If there are several boundary free vertices with the same priority, we remove the oldest vertex (the one that was added first to the cycle B). Intuitively, the goal of this approach is to retard the conquest of vertices having a small number of ingoing edges: observe that removing a vertex v having 0 ingoing edges leads to get $d-3$ edges of color 2 ingoing at v, while the number of ingoing edges of colors 0 and 1 would remain 0 (thus v would be unbalanced when its degree is greater than 4). From the implementation point of view, we use a collection of k queues $Q_0, Q_2, \ldots, Q_{k-1}$ (where k is a small constant), to store the vertices according to their priorities (the priority of vertices is stored and maintained using an integer array P of size n).

At the beginning of execution Q_0 contains only the outer vertex V_2 and the remaining queues are empty. When performing the vertex conquest of a free vertex v on B we add to Q_0 the neighboring vertices that are discovered (getting an outgoing black edge), while we increase the priority of v_l and v_r, the two left and right neighboring vertices of v, since they receive a new incoming edge (of color 0 and 1 respectively). We then add v_l and v_r at the



Tiny graphs	n	|F|	d_6	d_{max}	t_s	$\|\mathcal{S}(T)\|$
globe5	27	50	0.55	6	0	5 084 208
random	28	52	0.17	17	7	1294
Finger	28	52	0.21	8	2	38 760
delaunay	28	52	0.32	8	0	9 355 294
trigrid4	28	52	0.32	11	0	6 953 685
Hilbert	32	60	0.18	8	1	177 903
Desk	34	64	0.23	8	5	199 281
Lace	34	64	0.58	15	2	8 843 508

Figure 6.8: The table above reports some statistics for tiny triangulations: the proportion of degree 6 vertices, the maximum vertex degree and the number t_s of separating triangles. Last column reports the number $|\mathcal{S}(T)|$ of all distinct Schnyder woods of a given rooted triangulation.

balancedSchnyderWood($\mathcal{T}$, (V_0, V_1, V_2), k)
$B = \{V_0, V_1, V_2\}$ // *initialization*
$T =$ `new int[n]` // *priority array*
$Q_0 = \emptyset, Q_1 = \emptyset, \dots Q_{k-1} = \emptyset$ // *queue initialization*
Q_0`.addLast(`v_2`)`
`while(`$|B| \neq \{V_0, V_1\}$`) {`
  `let` M be the largest index s.t. $Q_M \neq \emptyset$
  `let` $v = Q_M$`.poll()`
  `if(`$v \in B$ `and` v `is free) {`
    `let` $\{v_l, v_{j_1}, \dots v_{j_t}, v_r\}$ be the neighbors of v on B
    `colorOrient(`v`)`
    `conquer(`v`)` // *remove v from B*
    $T[v_l]++, T[v_r]++$ // *increase priority*
    $Q_{\max(k-1, T[v_l])}$`.addLast(`v_l`)`
    $Q_{\max(k-1, T[v_r])}$`.addLast(`v_r`)`
    Q_0`.addLast(`v_{j_1}`), ..., `Q_0`.addLast(`v_{j_t}`)`
  `}`
`}`

Figure 6.7: Pseudo-code of the procedure for computing well balanced Schnyder woods.

end of $Q_{P[v_l]}$ and $Q_{P[v_r]}$ respectively. We do not increase the priority of a vertex having priority $k-1$ (thus having $k-1$ ingoing edges): the vertex is added to the end of Q_{k-1}. Observe that each vertex is added exactly once to Q_0, and can appear at most in $Q_0, Q_1, Q_2, \dots Q_r$, where $r = \min(k-1, deg(v)-3)$, since we have k queues, and a vertex receives $deg(v)$ incoming edges. At a given iteration of the incremental conquest, in order to chose a vertex candidate to be removed, we look for the largest index M for which Q_M is not empty: we remove the first vertex u from the front of Q_M and we check whether u is still on B and free (no incident chords), and we perform a `colorOrient` operation if this is the case.

RUNTIME PERFORMANCES. Our heuristic is very simple to implement and its linear-time behavior is confirmed by our experiments (where we set $k = 5$): the computation of balanced Schnyder woods is slightly slower than the computation of minimal Schnyder woods, but it is still extremely fast in practice, allowing us to process more than 3M vertices per second (on both real-world and synthetic triangulations).

POST-PROCESSING OPTIMIZATION: IMPROVING THE BALANCE. The balance of the Schnyder woods obtained via our heuristic can be further improved by performing the reversal of oriented triangles. To be more precise, one can iterate over all (cw or ccw) oriented triangle faces and check whether the reversal of the orientation leads to increase the number of balanced vertices incident to the triangle: if this occurs the reversal is performed leading to a Schnyder wood with slightly larger number of balanced vertices. As illustrated by the charts in Figures 6.9 and 6.10, the improvement one can achieve with this post-processing phase is rather important for regular graphs: in our experiments we observed that the proportion of balanced vertices (denoted δ_0) has increased between 14% and 88% on the tested meshes. The improvement is negligible for irregular (random) graphs. Unfortunately this post-processing phase is quite expensive: according to our experiments, iterating over the faces and reversing oriented triangles can be even 5.5 slower than running our heuristic.

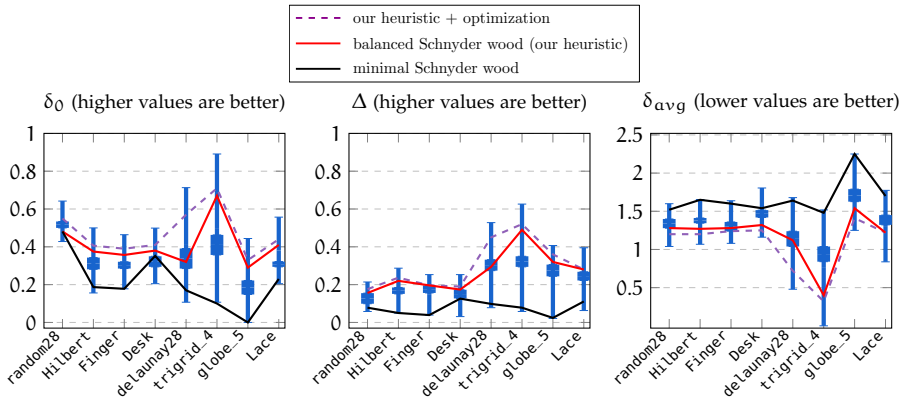


Figure 6.9: Balance of tiny graphs. Graphs are listed from left to right according to the increasing values of d_6 (most regular graphs are listed rightmost). Whisker plots (blue) represent the entire ranges of computed values for a given parameter over all distinct Schnyder woods.

6.2.2 *Experimental evaluation*

TINY GRAPHS. We consider tiny triangulations having less than 40 vertices, including both real-world and synthetic graphs (they are listed in Table 6.8): they are small enough so that we can generate all distinct Schnyder woods in a reasonable amount of time (a few hours) and compute some statistics involving their balance. More precisely, for a given rooted triangulation we compute all distinct Schnyder woods and plot three parameters (see charts in Fig. 6.9): the proportion δ_0 of *balanced* vertices, the *average vertex defect* δ_{avg} and the proportion Δ of *3-colored faces*. We use whisker plots (blue) to represent the entire ranges of computed values for a given parameter: boxes represent the middle 50% of values (25% – 75% percentile).

LARGE REAL-WORLD GRAPHS. To evaluate the balance quality of the Schnyder woods we plot the value δ_0 and Δ as a function of d_6: our balanced Schnyder woods are compared to minimal ones in Fig. 6.10. Experimental results strongly suggests that our heuristic leads to well balanced Schnyder woods. Our heuristic performs particularly well for regular graphs, for which a large majority of vertices are balanced (79% in average for the `sphere` graph). The results are good also for irregular graphs (`egea`), where about 45% of vertices are balanced. In our experiments we observed that the choice of the initial seed has a limited effect on the balance of the resulting Schnyder wood. Minimal Schnyder woods represent a bad case, especially for regular graphs: most vertices have a large defect and the resulting paths $\Pi_0(v)$, $\Pi_1(v)$ and $\Pi_2(v)$ resemble very long spirals.

6.3 APPLICATION I: SCHNYDER DRAWINGS

While recent works [LSW16] provide a probabilistic study of the converge for uniformly sampled triangulations endowed with a Schnyder wood, as far as we know there are no theoretical or empirical evaluations of the layout quality for regular graphs.

In this section our goal is to evaluate the impact of the balance of a Schnyder wood on the layout quality of the corresponding Schnyder drawings.

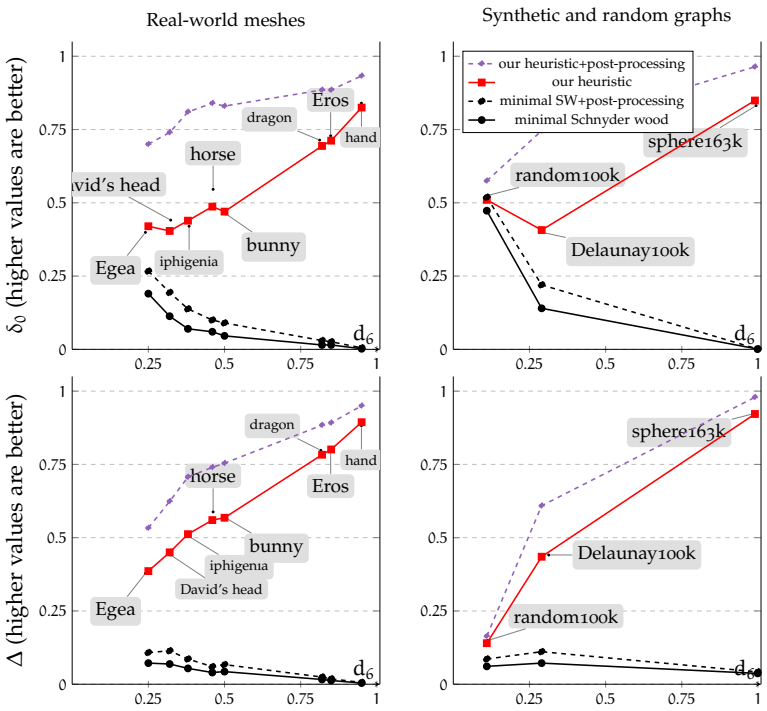


Figure 6.10: Balance of Schnyder woods for large graphs. For a given fixed choice of the initial seed we evaluate the proportion δ_0 of balanced vertices (top charts) and the proportion Δ of 3-colored faces (bottom charts): we compare our heuristic (red) to the balance of minimal Schnyder woods (black). Dashed charts represent the results obtained after running the post-processing phase. The results are expressed as functions of d_6 (most regular graphs appear rightmost).

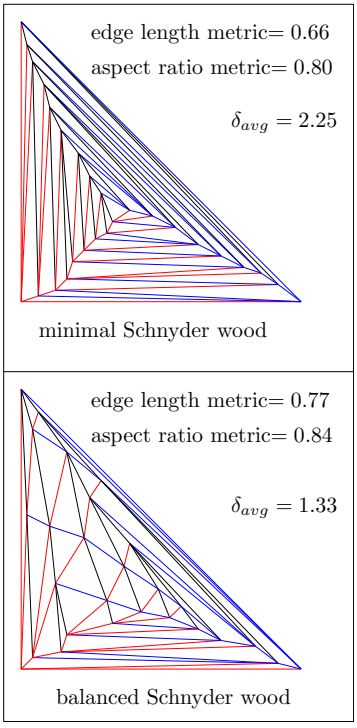


Figure 6.13: Schnyder drawings of the `globe_5` graph.

MOTIVATION. A first qualitative evaluation of the graph layouts based on the balance Schnyder woods is provided by the pictures in Figures 6.11 and 6.13, showing the layouts corresponding to distinct Schnyder woods of the same graph, which confirm the following intuition. In the case of regular graphs, when starting from our well balanced Schnyder woods the shape of triangles is much more balanced, and the resulting drawing partially captures the regularity of the grid. When starting from an unbalanced Schnyder wood the drawing exhibits many long edges and flat triangles, a typical drawback of Schnyder drawings.

MEASURING THE LAYOUT QUALITY. As a quantitative measure of the quality of a graph layout we consider two parameters: the edge length and the aspect ratio of the faces. More precisely, as already done in Section 5.7 we compute the *edge lengths aesthetic metric* $\mathfrak{el} = 1 - d_{el}$, where d_{el} is the average percent deviation of edge lengths defined by Equation 5.2 (obviously, in this chapter we consider the euclidean edge lengths). Observe that values of $\mathfrak{el}$ close to 1 mean that most edges have the same length, which is a desirable aesthetic criterion especially for regular graphs. We also use the *aspect ratio aesthetic metric*, denoted by $\mathfrak{ar}$, which can be defined in a similar way.

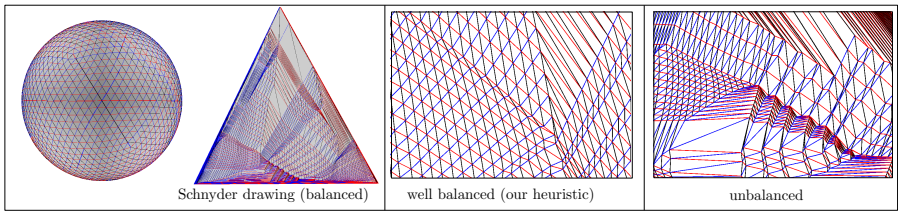


Figure 6.11: (left) A `sphere` graph endowed with our balanced Schnyder wood and the corresponding Schnyder drawing. The right pictures show the same portion of the Schnyder drawing: our balanced Schnyder wood leads to a much more readable and regular layout compared to the unbalanced case (right).

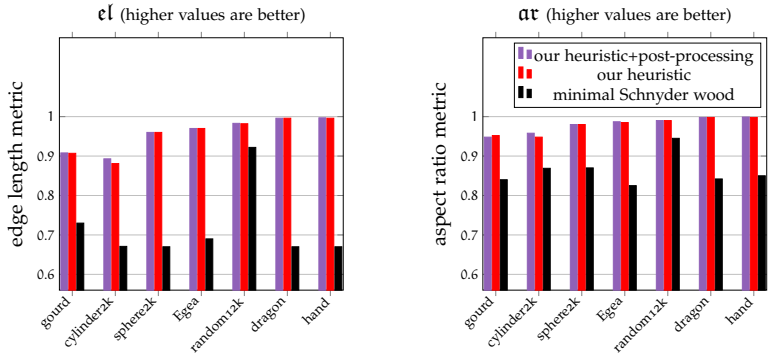


Figure 6.12: Layout quality of Schnyder drawings for medium and large graphs (smallest graphs are listed leftmost). For a fixed choice of the root face we compute a balanced Schnyder wood with our heuristic and the minimal one: we compare the layout quality of the corresponding Schnyder drawings.

EXPERIMENTAL EVALUATION The intuition, suggested by the layouts of Fig. 6.11 and Fig. 6.13, that balanced Schnyder woods lead to more pleasant and regular drawings, is also confirmed by the experiments involving several real-world and synthetic graphs. As shown by the charts in Fig. 6.12, the layouts corresponding to a balanced Schnyder woods always achieve better aesthetic criteria when compared with unbalanced Schnyder woods (in our tests we make use of the minimal one, which is strongly unbalanced).

To get a more quantitative evaluation, we consider some tiny graphs for which we are able to generate all possible Schnyder woods and compute the corresponding Schnyder drawings (with a fixed root face). Comparing the layout parameters we can state some interesting remarks: a more balanced Schnyder wood does not always necessarily guarantees to get a more pleasant Schnyder drawing, but this occurs very frequently. This is confirmed by our experiments shown in the charts of Fig. 6.14, reporting the values of several parameters (el, ar, as well as the average edge length and aspect ratio) as a function of the average defect (as a measure of the balance). More precisely, we compute all distinct Schnyder woods of a rooted triangulation and we partition them according to the parameter δ_{avg} as a measure of their balance. For every class of Schnyder woods achieving more or less the

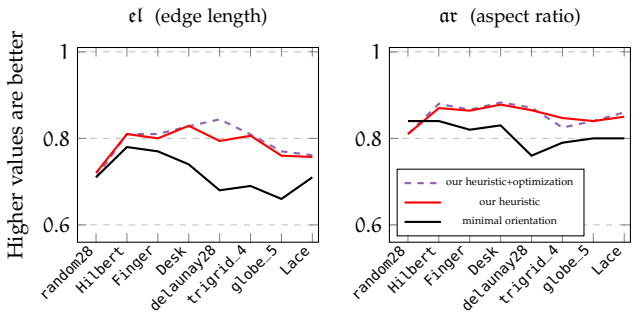


Figure 6.14: Comparison of layout quality: for each tiny graph we compute the edge length and aspect ratio metrics of the Schnyder drawings corresponding to our balanced Schnyder woods (compared to the drawing of a minimal Schnyder wood).

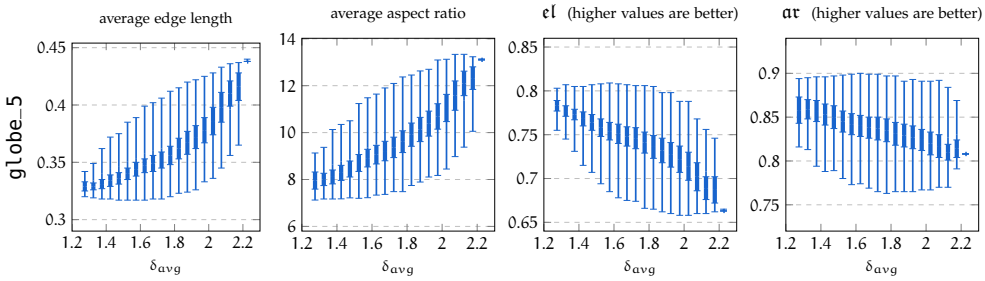


Figure 6.15: Layout statistics of Schnyder drawings. We generate all distinct Schnyder woods of the `globe_5` graph and we compute some statistics of the corresponding Schnyder drawings, according to the following aesthetic metrics: the average edge length (resp. aspect ratio) and the average percent deviation of edge Lent (resp. aspect ratio). We plot these statistics as a function of the average defect δ_{avg}: balanced Schnyder woods have smaller values of δ_{avg} and are listed leftmost: for the `globe_5` graph the values of δ_{avg} range in $[1.25, 2.5]$.

same balance, we make use of bar charts in order to plot the proportion of Schnyder drawings achieving a given value of the selected layout parameter. As shown by the plots in Fig. 6.15 for the case of the `globe_5` graph, the layout quality tends to deteriorate as soon as Schnyder woods get more unbalanced: Schnyder woods with high values of δ_{avg} are more likely to get worst layout statistics.

6.4 APPLICATION II: SMALL SEPARATORS

SMALL (CYCLE) SEPARATORS. Now we consider the problem of computing small separators, which has been extensively investigated during the last four decades, due to its relevance for many graph algorithms [LT79; LT80; Mil86; ST96]. Given a graph G we consider *small separators* which are defined by a partition (A, B, S) of all vertices such that S is a separating vertex set of small size (usually $|S| = O(\sqrt{n})$), and the remaining vertices in $G \setminus S$ belong to a balanced partition (A, B) satisfying $|A| \leqslant \alpha n$, $|B| \leqslant \alpha n$ (usually, for planar graphs, the *balance ratio* is $\alpha = \frac{2}{3}$). Here we focus on *simple cycle separators* [Mil86], for which fast implementations [Hol+09; Fox+16]

Despite the large number of existing algorithmic solutions involving graph separators, we have not been able to find any working (and publicly available) implementation of planar separator theorems.

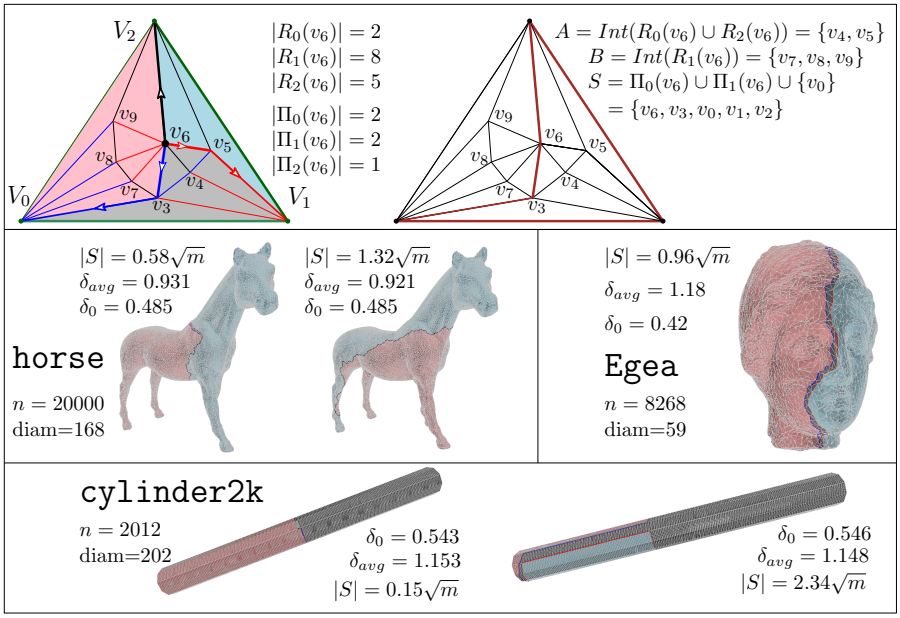
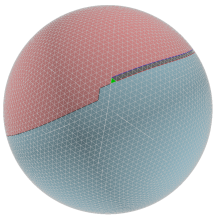


Figure 6.17: (top) A separator (A, B, S) obtained from a Schnyder wood. (bottom) Example of cycles separators computed via balanced Schnyder woods: the quality of the solution strongly depends on the choice of the initial seed (the root face).

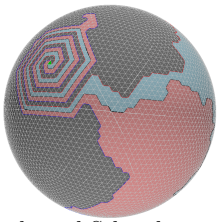
have been recently proposed (some of them [Fox+16] are provided with a worst-case bound of $\sqrt{8m}$ on the cycle size).

FROM SCHNYDER WOODS TO SIMPLE CYCLE SEPARATORS. Schnyder woods provides a very fast procedure for partitioning, given an arbitrary inner vertex v, the set of inner faces of a triangulation into three sets $R_0(v)$, $R_1(v)$ and $R_2(v)$ (respectively blue, red and gray triangles in Fig. 6.3(b)), whose boundaries consist of the three disjoint paths $\Pi_0(v)$, $\Pi_1(v)$ and $\Pi_2(v)$ emanating from v. The computation of simple cycle separators can be done as follows: for each vertex v check whether the two sets $A = Int(R_i(v) \cup R_{i+1}(v))$ and $B = Int(R_{i+2}(v))$ satisfy the prescribed balance ratio for at least one index $i \in \{0, 1, 2\}$ (indices are modulo 3, and $Int(R)$ denotes the set of inner vertices of a region R): then select the vertex for which the corresponding cycle length $|\Pi_i(v)| + |\Pi_{i+1}(v)| + 1$ is minimal (observe that, in practice, most vertices could lead to unbalanced partitions whose boundary size can be very large). All these steps can be performed almost instantaneously, since all the quantities above are encoded in the Schnyder drawing itself (see [Sch90] for more details). As far as we know there are no theoretical guarantees on both the partition balance and boundary size: in general, given an arbitrary Schnyder wood of a given triangulation, the procedure above does not lead to a short cycle separator (as depicted in Fig. 6.16). Nevertheless, our experiments seem to suggest that a balanced Schnyder wood is able to obtain good separators of short size for many classes of graphs; moreover, our approach is simple to implement and much faster than other existing solutions [Hol+09; Fox+16].

EXPERIMENTAL EVALUATION. In our experiments we look for separators with a balance ratio $\alpha = \frac{2}{3}$ that are *short*: the boundary size is at most $|S| \leqslant \sqrt{8m}$, as required in [Fox+16]. We plot in the charts of Fig. 6.18 the boundary sizes and partition balances of the separators obtained from a



balanced Schnyder wood



unbalanced Schnyder wood

Figure 6.16: The separators corresponding to a balanced and an unbalanced Schnyder woods of a regular sphere graph.

As observed in practice [Cas19], our `Java` *implementation allows processing between* 1.43M *and* 1.92M *vertices per second. This has to be compared to the* `C` *implementations of previous results on cycle separators [Hol+09; Fox+16], running on an Intel Xeon X5650 2.67GHz (with* 48.4 *GB of RAM): the fastest variant of the procedures tested in [Fox+16] allows processing between* 0.54M *and* 0.62M *vertices per second for the case of square* `grids`.

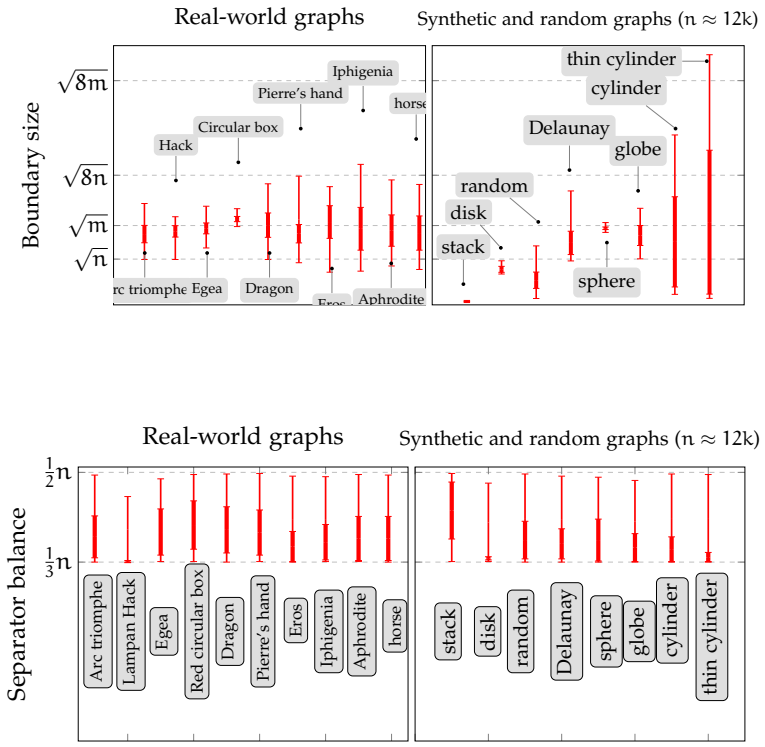


Figure 6.18: We evaluate the quality of the cycle separators obtained from our balanced Schnyder woods (tests are repeated with 200 random seeds). The top charts report the boundary sizes, while the bottom charts show the plots of the separator balance (the normalized size of the smallest of the two sets A and B). Graphs are listed from left to right according to the increasing values of their relative diameter.

Schnyder drawing as described in Section 6.2. Our tests, repeated over several tens of graphs, confirm our intuition that balanced Schnyder woods lead to good separators for a large majority of classes of graphs.

As one could expect, the separator size and balance strongly depend on the balance of the underlying Schnyder wood, and their are also affected by the choice of the seed for graphs with large diameter: a good choice of the seed would prevent from getting too long cycles. For graphs with small diameter (e.g. random triangulations) Schnyder woods lead to very short separators, while the size is closed to $\sqrt{m}$ for most real-world graphs. In order to be fair we have to mention that our separators are often longer when compared with the results obtained in [Fox+16], but well below the prescribed bound of $\sqrt{8m}$.

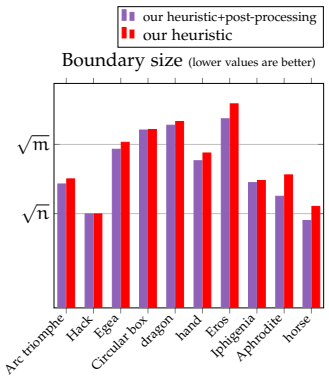


Figure 6.19: Separator quality: effect of post-processing optimization. For a fixed choice of the root face we compare the size of the separators obtained with our method, before and after performing the post-processing optimization.

IMPROVING THE SEPARATOR QUALITY VIA POST-PROCESSING. We already mentioned that it would be possible to perform a post-processing optimization in order to increase the balance of the Schnyder wood. As observed in our experiments the separator quality of the resulting Schnyder woods does not considerably improve after this post-processing phase: the plots in Fig. 6.19 show that the boundary size decreases less than 7% in most cases (on the tested meshes).

7 CONCLUDING REMARKS AND PERSPECTIVES

As shown throughout this manuscript Schnyder woods offer a very elegant and deep tool for dealing with graphs which are planar or embedded on surfaces, leading to extremely fast and memory efficient algorithmic solutions for solving several classes of graph problems. My impression is that Schnyder woods and related combinatorial structures have still many wonderful surprises in store for us. I list below a few open problems and lines of research that I find promising [1].

7.1 GRAPHS ON SURFACES: OPEN QUESTIONS

7.1.1 *Schnyder woods for graphs on surfaces*

As already mentioned in Sections 3.4.2 and 6.1, there are many interesting open questions in the higher genus case involving both the combinatorial structure and the algorithmics of Schnyder woods.

For the case of simple toroidal triangulations a crucial question [Lév16] is the existence of 3-orientations that satisfy the definition of toroidal Schnyder woods while preserving the global spanning condition (the graphs induced by the edges of the three colors define three connected subgraphs). To our knowledge this question is still open.

No doubt, one very challenging question involves the existence and computation of orientations satisfying the multiple Schnyder local rule for simple triangulations of genus $g \geqslant 2$: according to our experimental results (Section 6.1) we conjecture the existence of such orientations even for graphs having small edge-width.

Finally, it would be interesting to further explore the algorithmics underlying the vertex shelling procedure. Can one devise a new shelling procedure for computing toroidal Schnyder woods satisfying the crossing condition? Is it possible to generalize the shelling procedure in order to compute 3-orientations and Schnyder woods in the genus $g \geqslant 2$ case? A positive answer could have a huge impact in practice from the algorithmic point of view, since the current results [Lév16; Sua21] for toroidal and higher genus graphs are not provided with practical implementations.

7.1.2 *Drawing higher genus graphs (for $g > 1$)*

As mentioned in Chapter 5, both the region-counting principle underlying Schnyder drawings and the *shift* method of the FPP algorithm have been extended to the toroidal case leading to periodic crossing-free grid drawings.

1 Here I do not include my works involving graph encoding and visualizations problems for other classes of graphs, for which the kind of techniques described in this manuscript is not suitable. This occurs for instance in the case of graphs (such as complex networks) which do not exhibit a structure coming from a planar (or locally planar) embedding, and also in the case of graphs whose combinatorial structure can evolve over time (since the structure of Schnyder woods can dramatically change under local updates). In both cases these graphs require other kinds of mathematical tools and algorithmic techniques to deal with, and this goes beyond the scope of this manuscript.

It remains open to devise efficient combinatorial algorithms for computing periodic drawings of graphs embedded on surfaces of genus $g \geqslant 2$. This problem is much more challenging, since it is not known how to combine the recent results on higher genus orientations [AGK16; Sua21] with the algorithmic techniques mentioned above.

7.1.3 *Adaptive analysis: graph drawing and graph encoding*

The recent experimental results presented in Chapter 6 would suggest that the study of regularity of Schnyder woods could be a new interesting research direction.

No doubt, the dependence of the layout quality of Schnyder drawings on the balance of the corresponding Schnyder woods has been little investigated to date. One very interesting and challenging problem would be to provide a theoretical analysis of the behaviour observed in Section 6.3, which seems to suggest that a much more pleasing layout could be obtained with a balanced Schnyder wood.

A promising line of research comes from the graph encoding perspective: motivated by some recent preliminary results [2], we aim at designing new compression schemes and compact data structures which could be able to take profit of the regularity of triangle meshes. Our hope is to achieve better compression rates, while still guaranteeing rigorous upper bounds on the encoding size in the worst case. A crucial question concerns the choice of the size parameters (vertex degree distribution, proportion of degree 6 vertices, number of separating triangles, number of 3-colored faces, ...) to choose for establishing new adaptive bounds.

7.2 SCHNYDER WOODS FOR HIGHER DIMENSIONAL COMPLEXES

The study of Schnyder woods and related combinatorial structures in the case of higher dimensional complexes has been very little investigated so far [Eva+14; GI20]: this problem is fascinating and extremely challenging, mainly because the combinatorial structure of such objects is much richer and difficult to explore, compared to the case of planar (or local planar) graphs. One line of research could be to consider some nice classes of higher dimensional complexes (e.g. shellable, collapsible or vertex decomposable) which could be good candidates for generalizing some of the properties of Schnyder woods.

2 In a ongoing and unpublished work (collaboration with Olivier Devillers) we have been able to improve the upper bound of $4n$ references per vertex stated in Theorem 4.8, when the input triangulation is regular (large majority of degree 6 vertices).

BIBLIOGRAPHY

[AL15] Noam Aigerman and Yaron Lipman. "Orbifold Tutte embeddings." In: *ACM Trans. Graph.* 34.6 (2015), 190:1–190:12.

[AGK16] Boris Albar, Daniel Gonçalves, and Kolja Knauer. "Orienting Triangulations." In: *J. Graph Theory* 83.4 (2016), pp. 392–405. URL: https://doi.org/10.1002/jgt.22005.

[AH78] M. O. Albertson and J. P. Hutchinson. "On the independence ratio of a graph." In: *J. Graph. Theory* 2 (1978), pp. 1–8.

[Alexx] Marc Alexa. "Merging polyhedral shapes with scattered features." In: *The Visual Computer* 16.1 (2000), pp. 26–37.

[AG05] Pierre Alliez and Craig Gotsman. "Recent Advances in Compression of 3D Meshes." In: *Advances in Multiresolution for Geometric Modelling*. Springer, 2005, pp. 3–26. URL: http://dx.doi.org/10.1007/3-540-26808-1_1.

[AJ05] Tyler J Alumbaugh and Xiangmin Jiao. "Compact array-based mesh data structures." In: *Proceedings of the 14th International Meshing Roundtable*. Springer. 2005, pp. 485–503. DOI: 10.1007/3-540-29090-7_29.

[BT06] János Barát and Carsten Thomassen. "Claw-decompositions and tutte-orientations." In: *J. Graph Theory* 52.2 (2006), pp. 135–146. URL: https://doi.org/10.1002/jgt.20149.

[Bar+12] Jérémy Barbay, Luca Castelli Aleardi, Meng He, and J Ian Munro. "Succinct representation of labeled graphs." In: *Algorithmica* 62.1-2 (2012), pp. 224–257. DOI: 10.1007/s00453-010-9452-7. URL: https://hal.inria.fr/hal-00712915v1.

[BHJ09] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. "Gephi: An Open Source Software for Exploring and Manipulating Networks." In: *Proc. of the Third Int. Conf. on Weblogs and Social Media, ICWSM 2009, 2009*. 2009.

[BTV99] Giuseppe Di Battista, Roberto Tamassia, and Luca Vismara. "Output-Sensitive Reporting of Disjoint Paths." In: *Algorithmica* 23.4 (1999), pp. 302–340. URL: https://doi.org/10.1007/PL00009264.

[Bau72] Bruce G Baumgart. *Winged edge polyhedron representation*. Tech. rep. DTIC Document, 1972. URL: http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=AD0755141.

[Bau75] Bruce G Baumgart. "A polyhedron representation for computer vision." In: *Proceedings National Computer Conference and Exposition*. ACM. 1975, pp. 589–596. DOI: 10.1145/1499949.1500071.

[Ber07] O. Bernardi. "Bijective counting of tree-rooted maps and shuffle of parenthesis systems." In: *The Electronic Journal of Combinatorics* 14 (2007).

[BB07] O. Bernardi and N. Bonichon. "Catalans intervals and realizers of triangulations." In: *Proc. FPSAC07*. 2007.

[BB09] Olivier Bernardi and Nicolas Bonichon. "Catalan's intervals and realizers of triangulations." In: *Journal of Combinatorial Theory, Series A* 116.1 (2009). 22 pages, pp. 55–75. URL: https://hal.archives-ouvertes.fr/hal-00143870.

[BH18] Maciej Besta and Torsten Hoefler. "Survey and Taxonomy of Lossless Graph Compression and Space-Efficient Graph Representations." In: *CoRR* abs/1806.01799 (2018). arXiv: 1806.01799. URL: http://arxiv.org/abs/1806.01799.

[Boi+02] Jean-Daniel Boissonnat, Olivier Devillers, Sylvain Pion, Monique Teillaud, and Mariette Yvinec. "Triangulations in CGAL." In: *Computational Geometry: Theory & Applications* 22 (2002), pp. 5–19. DOI: 10.1016/S0925-7721(01)00054-2. URL: https://hal.inria.fr/inria-00167199.

[BGH03] N. Bonichon, C. Gavoille, and N. Hanusse. "An information-theoretic upper bound of planar graphs using triangulation." In: *STACS*. Springer, 2003, pp. 499–510.

[BGL05] N. Bonichon, C. Gavoille, and A. Labourel. "Edge partition of toroidal graphs into forests in linear time." In: *ICGT*. Vol. 22. 2005, pp. 421–425.

[Bon+06] N. Bonichon, C. Gavoille, N. Hanusse, D. Poulalhon, and G. Schaeffer. "Planar graphs, via well-orderly maps and trees." In: *Graphs and Combinatorics* 22.2 (2006), pp. 185–202.

[Bon02] Nicolas Bonichon. "Aspects algorithmiques et combinatoires des réaliseurs des graphes plans maximaux." PhD thesis. Bordeaux I, 2002.

[Bon+10a] Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and David Ilcinkas. "Connections between Theta-Graphs, Delaunay Triangulations, and Orthogonal Surfaces." In: *Graph Theoretic Concepts in Computer Science - 36th International Workshop, WG 2010*. 2010, pp. 266–278. URL: https://doi.org/10.1007/978-3-642-16926-7_25.

[Bon+10b] Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and Ljubomir Perkovic. "Plane Spanners of Maximum Degree Six." In: *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010*. 2010, pp. 19–30. URL: https://doi.org/10.1007/978-3-642-14165-2_3.

[BW00] U. Brandes and D. Wagner. "A Linear Time Algorithm for the Arc Disjoint Menger Problem in Planar Directed Graphs." In: *Algorithmica* 28.1 (2000), pp. 16–36. URL: https://doi.org/10.1007/s004530010029.

[Bre00] Enno Brehm. "3-orientations and Schnyder 3-tree-decompositions." In: *Master's Thesis, FB Mathematik und Informatik, Freie Universität Berlin* (2000).

[Bro64] W. Brown. "Enumeration of triangulations of the disk." In: *Proc. London Math. Soc.* 3.14 (1964), pp. 515–528.

[CM07] Sergio Cabello and Bojan Mohar. "Finding Shortest Non-Separating and Non-Contractible Cycles for Topologically Embedded Graphs." In: *Discrete & Comp. Geometry* 37.2 (2007), pp. 213–235.

[CKS98] Swen Campagna, Leif Kobbelt, and Hans-Peter Seidel. "Directed Edges—A Scalable Representation for Triangle Meshes." In: *Journal of Graphics Tools* 3 (1998), pp. 1–11. DOI: 10.1080/10867651.1998.10487494.

[Cas19] Luca Castelli Aleardi. "Balanced Schnyder Woods for Planar Triangulations: An Experimental Study with Applications to Graph Drawing and Graph Separators." In: *Graph Drawing and Network Visualization - 27th International Symposium, GD*. Vol. 11904. Lecture Notes in Computer Science. Springer, 2019, pp. 114–121. URL: https://doi.org/10.1007/978-3-030-35802-0_9.

[CDF18a] Luca Castelli Aleardi, Gaspard Denis, and Éric Fusy. "Fast Spherical Drawing of Triangulations: An Experimental Study of Graph Drawing Tools." In: *17th International Symposium on Experimental Algorithms, SEA*. Vol. 103. LIPIcs. 2018, 24:1–24:14. URL: https://hal.archives-ouvertes.fr/hal-01761754.

[CD18] Luca Castelli Aleardi and Olivier Devillers. "Array-based compact data structures for triangulations: Practical solutions with theoretical guarantees." In: *J. Comput. Geom.* 9.1 (2018), pp. 247–289. URL: https://doi.org/10.20382/jocg.v9i1a8.

[CDF12] Luca Castelli Aleardi, Olivier Devillers, and Éric Fusy. "Canonical Ordering for Triangulations on the Cylinder, with Applications to Periodic Straight-Line Drawings." In: *Graph Drawing - 20th International Symposium*. 2012, pp. 376–387.

[CDF18b] Luca Castelli Aleardi, Olivier Devillers, and Éric Fusy. "Canonical ordering for graphs on the cylinder, with applications to periodic straight-line drawings on the flat cyclinder and torus." In: *J. Comput. Geom.* 9.1 (2018), pp. 391–429. URL: https://doi.org/10.20382/jocg.v9i1a14.

[CDM11] Luca Castelli Aleardi, Olivier Devillers, and Abdelkrim Mebarki. "Catalog Based Representation of 2D triangulations." In: *International Journal of Computational Geometry & Applications* 21 (2011), pp. 393–402. DOI: 10.1142/S021819591100372X. URL: http://hal.inria.fr/inria-00560400.

[CADR12] Luca Castelli Aleardi, Olivier Devillers, and Jarek Rossignac. "ESQ: Editable SQuad Representation for Triangle Meshes." In: *25th Conference on Graphics, Patterns and Images, SIBGRAPI 2012*. 2012, pp. 110–117. URL: http://dx.doi.org/10.1109/SIBGRAPI.2012.24.

[CDS05a] Luca Castelli Aleardi, Olivier Devillers, and Gilles Schaeffer. "Dynamic updates of succinct triangulations." In: *Proceedings of the 17th Canadian Conference on Computational Geometry*. 2005, pp. 134–137. URL: http://www.cccg.ca/proceedings/2005/45.pdf.

[CDS05b] Luca Castelli Aleardi, Olivier Devillers, and Gilles Schaeffer. "Succinct Representation of Triangulations with a Boundary." In: *Algorithms and Data Structures, 9th International Workshop, WADS*. Vol. 3608. Lecture Notes in Computer Science. Springer, 2005, pp. 134–145. URL: https://doi.org/10.1007/11534273_13.

[CDS08] Luca Castelli Aleardi, Olivier Devillers, and Gilles Schaeffer. "Succinct representations of planar maps." In: *Theor. Comput. Sci.* 408.2-3 (2008), pp. 174–187. URL: https://doi.org/10.1016/j.tcs.2008.08.016.

[CFK14] Luca Castelli Aleardi, Éric Fusy, and Anatolii Kostrygin. "Periodic Planar Straight-Frame Drawings with Polynomial Resolution." In: *LATIN 2014: Theoretical Informatics - 11th Latin American Symposium*. 2014, pp. 168–179.

[CAFL09] Luca Castelli Aleardi, Éric Fusy, and Thomas Lewiner. "Schnyder woods for higher genus triangulated surfaces, with applications to encoding." In: *Discrete & Computational Geometry* 42.3 (2009), pp. 489–516. DOI: 10.1007/s00454-009-9169-z. URL: https://hal.inria.fr/hal-00712046v1.

[CFL10] Luca Castelli Aleardi, Éric Fusy, and Thomas Lewiner. "Optimal encoding of triangular and quadrangular meshes with fixed topology." In: *Proceedings of the 22nd Annual Canadian Conference on Computational Geometry*. 2010, pp. 95–98. URL: http://cccg.ca/proceedings/2010/paper27.pdf.

[Cha+12] Erin W. Chambers, David Eppstein, Michael T. Goodrich, and Maarten Löffler. "Drawing Graphs in the Plane with a Prescribed Outer Face and Polynomial Area." In: *J. Graph Algorithms Appl.* 16.2 (2012), pp. 243–259.

[CP95] M. Chrobak and T. H. Payne. "A Linear-Time Algorithm for Drawing a Planar Graph on a Grid." In: *Inf. Process. Lett.* 54.4 (1995), pp. 241–246.

[Chu+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: *Automata, Languages and Programming*. Springer, 1998, pp. 118–129. DOI: 10.1007/BFb0055046.

[DFM01] Hubert De Fraysseix and Patrice Ossona de Mendez. "On topological aspects of orientations." In: *Discrete Mathematics* 229.1 (2001), pp. 57–72. DOI: 10.1016/S0012-365X(00)00201-6.

[DGL17] Vincent Despré, Daniel Gonçalves, and Benjamin Lévêque. "Encoding Toroidal Triangulations." In: *Discrete & Computational Geometry* 57.3 (2017), pp. 507–544. URL: https://doi.org/10.1007/s00454-016-9832-0.

[Dha10] Raghavan Dhandapani. "Greedy Drawings of Triangulations." In: *Discrete & Computational Geometry* 43.2 (2010), pp. 375–392. URL: https://doi.org/10.1007/s00454-009-9235-6.

[DGK11] Christian A. Duncan, Michael T. Goodrich, and Stephen G. Kobourov. "Planar Drawings of Higher-Genus Graphs." In: *J. Graph Algorithms Appl.* 15.1 (2011), pp. 7–32.

[Ell+01] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. "Graphviz - Open Source Graph Drawing Tools." In: *Proc. of Graph Drawing*. 2001, pp. 483–484.

[Eva+14] William Evans, Stefan Felsner, Stephen G. Kobourov, and Torsten Ueckerdt. "Graphs admitting d-realizers: spanning-tree-decompositions and box-representations." In: *Proc. of EuroCG*. 2014.

[Fel04] S. Felsner. "Lattice structures from planar graphs." In: *Electronic Journal of Combinatorics* 11.15 (2004), p. 24.

[Fel01] Stefan Felsner. "Convex drawings of planar graphs and the order dimension of 3-polytopes." In: *Order* 18.1 (2001), pp. 19–37.

[Fel03] Stefan Felsner. "Geodesic Embeddings and Planar Graphs." In: *Order* 20.2 (2003), pp. 135–150. URL: https://doi.org/10.1023/B:ORDE.0000009251.68514.8b.

[FK12] J. Joseph Fowler and Stephen G. Kobourov. "Planar Preprocessing for Spring Embedders." In: *Graph Drawing - 20th International Symposium*. 2012, pp. 388–399.

[Fox+16] Eli Fox-Epstein, Shay Mozes, Phitchaya Mangpo Phothilimthana, and Christian Sommer. "Short and Simple Cycle Separators in Planar Graphs." In: *ACM Journal of Experimental Algorithmics* 21.1 (2016), 2.2:1–2.2:24.

[FPP90] Hubert de Fraysseix, János Pach, and Richard Pollack. "How to draw a planar graph on a grid." In: *Combinatorica* 10.1 (1990), pp. 41–51.

[FR91] Thomas M. J. Fruchterman and Edward M. Reingold. "Graph Drawing by Force-directed Placement." In: *Softw., Pract. Exper.* 21.11 (1991), pp. 1129–1164.

[FPS05] E. Fusy, D. Poulalhon, and G. Schaeffer. "Dissections and trees, with applications to optimal mesh encoding and to random sampling." In: *SODA*. 2005, pp. 690–699.

[Fus07] Eric Fusy. "Combinatoire des cartes planaires et applications algorithmiques." PhD thesis. Ecole Polytechnique, 2007.

[Gao93] Z. Gao. "A pattern for the asymptotic number of rooted maps on surfaces." In: *Journal of Combinatorial Theory, Series A* 64 (1993), pp. 246–264.

[GI20] Daniel Gonçalves and Lucas Isenmann. "Dushnik-Miller dimension of TD-Delaunay complexes." In: *Eur. J. Comb.* 88 (2020), p. 103110. URL: https://doi.org/10.1016/j.ejc.2020.103110.

[GL14] Daniel Gonçalves and Benjamin Lévêque. "Toroidal Maps: Schnyder Woods, Orthogonal Surfaces and Straight-Line Representations." In: *Discrete & Computational Geometry* 51.1 (2014), pp. 67–131. URL: http://dx.doi.org/10.1007/s00454-013-9552-7.

[GGT06] Steven J. Gortler, Craig Gotsman, and Dylan Thurston. "Discrete one-forms on meshes and applications to 3D mesh parameterization." In: *Comput. Aided Geom. Des.* 23.2 (2006), pp. 83–112. URL: https://doi.org/10.1016/j.cagd.2005.05.002.

[Got03] Craig Gotsman. "On the Optimality of Valence-based Connectivity Coding." In: *Comput. Graph. Forum* 22.1 (2003), pp. 99–102. URL: https://doi.org/10.1111/1467-8659.t01-1-00649.

[GGS03] Craig Gotsman, Xianfeng Gu, and Alla Sheffer. "Fundamentals of spherical parameterization for 3D meshes." In: *ACM Trans. Graph.* 22.3 (2003), pp. 358–363.

[GS85] Leonidas Guibas and Jorge Stolfi. "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams." In: *ACM transactions on graphics (TOG)* 4.2 (1985), pp. 74–123. DOI: 10.1145/282918.282923.

[GR09] Topraj Gurung and Jarek Rossignac. "SOT: compact representation for tetrahedral meshes." In: *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*. ACM. 2009, pp. 79–88. DOI: 10.1145/1629255.1629266.

[Gur+11a] Topraj Gurung, Mark Luffel, Peter Lindstrom, and Jarek Rossignac. "LR: compact connectivity representation for triangle meshes." In: *ACM transactions on graphics (TOG)* 30.4 (2011). DOI: 10.1145/2010324.1964962.

[Gur+11b] Topraj Gurung, Daniel Laney, Peter Lindstrom, and Jarek Rossignac. "SQuad: Compact representation for triangle meshes." In: *Computer Graphics Forum* 30.2 (2011), pp. 355–364. DOI: 10.1111/j.1467-8659.2011.01866.x.

[Gur+13] Topraj Gurung, Mark Luffel, Peter Lindstrom, and Jarek Rossignac. "Zipper: A compact connectivity data structure for triangle meshes." In: *Computer-Aided Design* 45.2 (2013), pp. 262–269. URL: http://dx.doi.org/10.1016/j.cad.2012.10.009.

[HKL99] Xin He, Ming-Yang Kao, and Hsueh-I Lu. "Linear-Time Succinct Encodings of Planar Graphs via Canonical Orderings." In: *SIAM J. Discrete Math.* 12.3 (1999), pp. 317–325. URL: http://dx.doi.org/10.1137/S0895480197325031.

[Hol+09] Martin Holzer, Frank Schulz, Dorothea Wagner, Grigorios Prasinos, and Christos D. Zaroliagis. "Engineering planar separator algorithms." In: *ACM Journal of Experimental Algorithmics* 14 (2009). URL: https://doi.org/10.1145/1498698.1571635.

[IS05] M. Isenburg and J. Snoeyink. *Graph Coding and Connectivity Compression*. 2005. URL: http://www.cs.unc.edu/~isenburg/papers/is-gccc-04.pdf.

[KT01] Marcelo Kallmann and Daniel Thalmann. "Star-vertices: a compact representation for planar meshes with adjacency information." In: *Journal of Graphics Tools* 6.1 (2001), pp. 7–18. DOI: 10.1080/10867651.2001.10487533.

[Kan96] Goos Kant. "Drawing Planar Graphs Using the Canonical Ordering." In: *Algorithmica* 16.1 (1996), pp. 4–32. URL: https://doi.org/10.1007/BF02086606.

[Ket99] Lutz Kettner. "Using generic programming for designing a data structure for polyhedral surfaces." In: *Comput. Geom.* 13.1 (1999), pp. 65–90. URL: https://doi.org/10.1016/S0925-7721(99)00007-3.

[KR99] Davis King and Jarek Rossignac. "Guaranteed 3.67v bit encoding of planar triangle graphs." In: *Proc. of the 11th Canadian Conf. on Comp. Geom.* 1999. URL: http://www.cccg.ca/proceedings/1999/c39.pdf.

[KGL19] Kolja Knauer, Daniel Gonçalves, and Benjamin Lévêque. "On the structure of Schnyder woods on orientable surfaces." In: *J. Comput. Geom.* 10.1 (2019), pp. 127–163. URL: https://doi.org/10.20382/jocg.v10i1a5.

[Kob13] Stephen G. Kobourov. "Force-Directed Drawing Algorithms." In: *Handbook on Graph Drawing and Visualization.* 2013, pp. 383–408.

[KW05] Stephen G. Kobourov and Kevin Wampler. "Non-Euclidean Spring Embedders." In: *IEEE Trans. Vis. Comput. Graph.* 11.6 (2005), pp. 757–767.

[Kut06] Martin Kutz. "Computing shortest non-trivial cycles on orientable surfaces of bounded genus in almost linear time." In: *SoCG*. 2006, pp. 430–438.

[Laz+01] Francis Lazarus, Michel Pocchiola, Gert Vegter, and Anne Verroust. "Computing a canonical polygonal schema of an orientable triangulated surface." In: *SoCG*. 2001, pp. 80–89.

[Lév16] Benjamin Lévêque. *Generalization of Schnyder woods to orientable surfaces and applications*. 2016. URL: https://tel.archives-ouvertes.fr/tel-01488943.

[LLT03] Thomas Lewiner, Hélio Lopes, and Geovan Tavares. "Optimal discrete Morse functions for 2-manifolds." In: *Comput. Geom.* 26.3 (2003), pp. 221–233. URL: https://doi.org/10.1016/S0925-7721(03)00014-2.

[LSW16] Yiting Li, Xin Sun, and Samuel S. Watson. *Schnyder woods, SLE(16), and Liouville quantum gravity*. Tech. rep. arXiv:1705.03573v1 [math.PR]. ArXiV, 2016. URL: https://arxiv.org/abs/1705.03573.

[LT79] Richard J. Lipton and Robert E Tarjan. "A Separator Theorem for Planar Graphs." In: *SIAM J. Applied Math.* 36.2 (1979), pp. 177–189.

[LT80] Richard J. Lipton and Robert Endre Tarjan. "Applications of a Planar Separator Theorem." In: *SIAM J. Comput.* 9.3 (1980), pp. 615–627. URL: https://doi.org/10.1137/0209046.

[Lis99] Valery A. Liskovets. "A Pattern of Asymptotic Vertex Valency Distributions in Planar Maps." In: *J. Comb. Theory, Ser. B* 75.1 (1999), pp. 116–133. URL: https://doi.org/10.1006/jctb.1998.1870.

[Lop+03] Hélio Lopes, Jarek Rossignac, Alla Safonova, Andrzej Szymczak, and Geovan Tavares. "Edgebreaker: a simple implementation for surfaces with handles." In: *Computers & Graphics* 27.4 (2003), pp. 553–567.

[Mag+15] Adrien Maglo, Guillaume Lavoué, Florent Dupont, and Céline Hudelot. "3D Mesh Compression: Survey, Comparisons, and Emerging Trends." In: *ACM Comput. Surv.* 47.3 (2015), 7:1–7:41. URL: http://dx.doi.org/10.1145/2693443.

[McD08] Colin McDiarmid. "Random graphs on surfaces." In: *J. Comb. Theory, Ser. B* 98.4 (2008), pp. 778–797. URL: https://doi.org/10.1016/j.jctb.2007.11.006.

[Men94] Patrice Ossona de Mendez. "Orientations bipolaires." PhD thesis. Paris, 1994.

[Mil86] Gary L. Miller. "Finding Small Simple Cycle Separators for 2-Connected Planar Graphs." In: *J. Comput. Syst. Sci.* 32.3 (1986), pp. 265–279. URL: https://doi.org/10.1016/0022-0000(86)90030-9.

[Moh96] Bojan Mohar. "Straight-line representations of maps on the torus and other flat surfaces." In: *Discret. Math.* 155.1-3 (1996), pp. 173–181. URL: https://doi.org/10.1016/0012-365X(94)00381-R.

[MR98] Bojan Mohar and Pierre Rosenstiehl. "Tessellation and Visibility Representations of Maps on the Torus." In: *Discret. Comput. Geom.* 19.2 (1998), pp. 249–263. URL: https://doi.org/10.1007/PL00009344.

[MT01] Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. Johns Hopkins, 2001.

[NSI04] Hiroshi Nagamochi, Takahisa Suzuki, and Toshimasa Ishii. "A simple recognition of maximal planar graphs." In: *Inf. Process. Lett.* 89.5 (2004), pp. 223–226.

[NR04] Takao Nishizeki and Md. Saidur Rahman. *Planar Graph Drawing*. Vol. 12. Lecture Notes Series on Computing. World Scientific, 2004. URL: https://doi.org/10.1142/5648.

[PS06] Dominique Poulalhon and Gilles Schaeffer. "Optimal coding and sampling of triangulations." In: *Algorithmica* 46.3-4 (2006), pp. 505–527. DOI: 10.1007/s00453-006-0114-8.

[RS86] Neil Robertson and Paul D. Seymour. "Graph minors. VI. Disjoint paths across a disc." In: *J. Comb. Theory, Ser. B* 41.1 (1986), pp. 115–138. URL: https://doi.org/10.1016/0095-8956(86)90031-6.

[Ros99] Jarek Rossignac. "Edgebreaker: Connectivity compression for triangle meshes." In: *Visualization and Computer Graphics, IEEE Transactions on* 5.1 (1999), pp. 47–61. DOI: 10.1109/2945.764870.

[Sab+05] S. Saba, I. Yavneh, C. Gotsman, and A. Sheffer. "Practical Spherical Embedding of Manifold Triangle Meshes." In: *(SMI2005)*. 2005, pp. 258–267.

[Sch89] Walter Schnyder. "Planar graphs and poset dimension." In: *Order* (1989), pp. 323–343.

[Sch90] Walter Schnyder. "Embedding planar graphs on the grid." In: *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*. Vol. 90. 1990, pp. 138–148. URL: http://departamento.us.es/dma1euita/PAIX/Referencias/schnyder.pdf.

[SS99] Jack Snoeyink and Bettina Speckmann. "Tripod: a minimalist data structure for embedded triangulations." In: *Workshop on Comput. Graph Theory and Combinatorics*. 1999. URL: https://www.win.tue.nl/~speckman/papers/Tripod.pdf.

[ST96] Daniel A. Spielman and Shang-Hua Teng. "Disk Packings and Planar Separators." In: *Proc. of the Twelfth Annual Symposium on Computational Geometry, 1996*. 1996, pp. 349–358.

[Sua21] Jason Suagee. "Existence and Construction of Schnyder Orientations over a Large Class of Higher Genus Surface Triangulations." PhD thesis. George Washington University, 2021.

[Sul17] Thom Sulanke. "Generating Maps on Surfaces." In: *Discret. Comput. Geom.* 57.2 (2017), pp. 335–356. URL: https://doi.org/10.1007/s00454-016-9853-8.

[SL09] Thom Sulanke and Frank H. Lutz. "Isomorphism-free lexicographic enumeration of triangulated surfaces and 3-manifolds." In: *Eur. J. Comb.* 30.8 (2009), pp. 1965–1979. URL: https://doi.org/10.1016/j.ejc.2008.12.016.

[Tar74] R.E. Tarjan. "A note on finding the bridges of a graph." In: *Inf. Process. Lett.* 2 (1974), pp. 160–161.

[Tur84] G. Turan. "On the succinct representation of graphs." In: *Discrete & Applied Mathematics* 8 (1984), pp. 289–294.

[Tut62] W. Tutte. "A census of planar triangulations." In: *Canadian Journal of Mathematics* 14 (1962), pp. 21–38.

[Tut63] W. Tutte. "How to draw a graph." In: *Proc. of London Math. Soc.* 13 (1963), pp. 734–767.

[VY90] Gert Vegter and Chee-Keng Yap. "Computational Complexity of Combinatorial Surfaces." In: *SoCG*. 1990, pp. 102–111.

[WKS01] D. Neilson W. Kocay and R. Szypowski. "Drawing graphs on the torus." In: *Ars Combinatoria* 59 (2001), pp. 259–277.

[Wal03] Chris Walshaw. "A Multilevel Algorithm for Force-Directed Graph-Drawing." In: *J. Graph Algorithms Appl.* 7.3 (2003), pp. 253–285.

[ZRS06] Rhaleb Zayer, Christian Rössl, and Hans-Peter Seidel. "Curvilinear Spherical Parameterization." In: *Int. Conf. on Shape Modeling and Applications (SMI 2006)*. 2006, p. 11.

[ZJ16] Qingnan Zhou and Alec Jacobson. "Thingi10K: A Dataset of 10, 000 3D-Printing Models." In: *CoRR* abs/1605.04797 (2016). URL: http://arxiv.org/abs/1605.04797.

[Zit94] Arjana Zitnik. "Drawing Graphs on Surfaces." In: *SIAM J. Discret. Math.* 7.4 (1994), pp. 593–597. URL: https://doi.org/10.1137/S0895480192242006.