National and Kapodistrian University of Athens
Department of Methematics

Graduate Program in Logic, Algorithms
and Computation

Triangulation Problems on Geometric Graphs – Sampling over Convex Triangulations

M.Sc. Thesis

of

Alexandros Angelopoulos

Supervisor: Aris Pagourtzis
Associate Professor N.T.U.A.

Thesis Committee complemented by: Stathis Zachos
Professor Emeritus N.T.U.A.

Dimitris Fotakis
Assistant Professor N.T.U.A.

Athens, November 2017

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Τμήμα Μαθηματικών

Μεταπτυχιακό Πρόγραμμα Λογικής, Θεωρίας Αλγορίθμων
και Υπολογισμού

ΠΡΟΒΛΗΜΑΤΑ ΤΡΙΓΩΝΟΠΟΙΗΣΗΣ ΣΕ ΓΕΩΜΕΤΡΙΚΑ ΓΡΑΦΗΜΑΤΑ – ΔΕΙΓΜΑΤΟΛΗΨΙΑ ΚΥΡΤΩΝ ΤΡΙΓΩΝΟΠΟΙΗΣΕΩΝ

Διπλωματική Εργασία

του

Αλέξανδρου Αγγελόπουλου

Επιβλέπων:	Αριστείδης Παγουρτζής Αναπληρωτής Καθηγητής Ε.Μ.Π.
Την Εξεταστική Επιτροπή συμπληρώνουν οι:	Ευστάθιος Ζάχος Ομότιμος Καθηγητής Ε.Μ.Π.
	Δημήτρης Φωτάκης Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2017

Η παρούσα Διπλωματική Εργασία

εκπονήθηκε στο πλαίσιο των σπουδών

για την απόκτηση του

Μεταπτυχιακού Διπλώματος Ειδίκευσης

στη

Λογική και Θεωρία Αλγορίθμων και Υπολογισμού

που απονέμει το

Τμήμα Μαθηματικών

του

Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών

Εγκρίθηκε την 10/11/2017 από την Εξεταστική Επιτροπή αποτελούμενη από τους:

Ονοματεπώνυμο	Βαθμίδα	Υπογραφή
Ευστάθιος Ζάχος	Ομότιμος Καθηγητής Ε.Μ.Π	....
Αριστείδης Παγουρτζής	Αναπληρωτής Καθηγητής Ε.Μ.Π.	....
Δημήτρης Φωτάκης	Επίκουρος Καθηγητής Ε.Μ.Π.	....

Abstract

A geometric graph is a set of points V on the plane and a set of straight line segments E with endpoints in V, potentially and instinctively associated with the abstract $G(V,E)$. When studying its thickness, i.e. partitioning its edges into crossing-free subsets (an **NP**-hard optimization problem), the problem of *triangulation existence* as a crossing-free subset T of the edges naturally occurs, as a triangulation of V is the largest such possible set that may be defined on V. In this Thesis, we examine a family of triangulation existence problems and classify them with respect to their complexity, both for their *decision* and their *counting* versions. The *general case decision problem* is the only one appearing in bibliography (Lloyd, 1977, **NP**-hard), while we deal with the *convex case* restriction and an "intermediate" *polygon triangulation existence* problem, fixing a new 2 by 2 table of results. In the final chapter, we modify our framework in order to build an exact uniform sampling and optimal coding algorithm for convex triangulations, which outperforms any known algorithm to date.

Keywords

geometric graph, triangulation existence, complexity, counting complexity, convex triangulations sampling

Περίληψη

Γεωμετρικό γράφημα καλείται ένα σύνολο σημείων V στο επίπεδο μαζί με ένα σύνολο ευθυγράμμων τμημάτων (ακμών) E που έχουν τα άκρα τους στο V, και εύκολα συσχετίζεται με τον «αφηρημένο» γράφημα $G(V, E)$. Μελετώντας το πάχος του, δηλαδή τη διαμέριση των ακμών του σε υποσύνολα ελεύθερα διασταυρώσεων (ένα **NP**-δύσκολο πρόβλημα βελτιστοποίησης), προκύπτει και το πρόβλημα της *ύπαρξης τριγωνοποίησης* ως ένα ελεύθερο διασταυρώσεων υποσύνολο T των ακμών, καθώς μια τριγωνοποίηση του V αποτελεί το μέγιστο δυνατό τέτοιο σύνολο που είναι δυνατόν να οριστεί δεδομένου του V. Η Διπλωματική αυτή Εργασία αφορά στη μελέτη μιας οικογένειας προβλημάτων ύπαρξης τριγωνοποίησης και την ταξινόμησή τους ως προς την πολυπλοκότητα *απόφασης*, αλλά και *μέτρησης*. Από αυτά, *το γενικό πρόβλημα απόφασης* είναι το μόνο μελετημένο στη βιβλιογραφία (Lloyd, 1977, **NP**-δύσκολο), ενώ εμείς μελετάμε αφ' ενός την ειδική περίπτωση των *κυρτών γεωμετρικών γραφημάτων*, αφ' ετέρου ένα «ενδιάμεσο» πρόβλημα *ύπαρξης τριγωνοποιημένου πολυγώνου*, δημιουργώντας έναν νέο 2×2 πίνακα αποτελεσμάτων. Στο τελευταίο κεφάλαιο, τροποποιούμε το πλαίσιο της δουλειάς μας έτσι ώστε να κατασκευάσουμε έναν αλγόριθμο για ομοιόμορφη δειγματοληψία και βέλτιστη κωδικοποίηση των κυρτών τριγωνοποιήσεων, ο οποίος υπερέχει έναντι κάθε γνωστού αλγορίθμου έως σήμερα.

Λέξεις-κλειδιά

γεωμετρικό γράφημα, ύπαρξη τριγωνοποίησης, πολυπλοκότητα, πολυπλοκότητα μέτρησης, δειγματοληψία κυρτών τριγωνοποιήσεων

Preface

I consider this work to be a first extension of my Diploma Thesis (NTUA, 2012, [3]). The *triangulation existence* problems were formulated back then, as side problems to the main objective I had in mind. The decision problems were solved quite quickly, however, there has not been any attempt to publish these results, as I would have liked to solidify their importance by showing more about their nature and relation to known problems.

The latest was achieved (in my opinion) with a little help by *counting complexity*. This a go-to field for Corelab, due to Stahis Zachos' and Aris Pagourtzis' work; more importantly, it proved to be a creative way both for better exploration of the triangulation problems, plus for getting me involved in an area I initially thought it does not "speak" to me. *Chapters 3 and 4* are a concrete impression of the above, the *results of which are joint work with Eleni Mpakali*, PhD student of Stathis Zachos.

A note on the writing style. There are definitions and proofs for which I omit any reference, while I am certain, or even have seen they do appear in bibliography. My excuse lies on at least one of the following: they are simple and easy to verify their correctness; maintaining a slightly altered phrasing, notation or terminology works in favor of reading seamlessly through the document.

During adulthood, one must master a significant number of various decisions
And then have to live with them
I hereby acknowledge the importance of people
Who try to be helpful *whilst respecting their mate's personal axioms*
Lucky to know some of those
They are labeled: *the close-ones*
And **I dedicate** the present bit of achievement to them

Contents

List of Figures

1 Introduction

1.1 Motivation

The question whether a graph is planar is one of the most well-studied problems in Graph Theory. Though it is not difficult to recognize a planar graph [21] and eventually draw it without crossing edges, it opens up more difficult questions, regarding how non-planar graphs can be drawn on $\mathbb{R}^2$ as to better understand and work on them. So, Graph Drawing is born naturally, with numerous goals, among which is to optimize graph visualization and VLSI design. In [3] we give an additional motivation, that of air traffic separation within congested airspace, and actually approach in a slightly different way: we are given the drawing of some graph and raise questions on the *geometric graph* itself. Some sample questions, given a geometric graph are the following:

- What is the *optimal edge partitioning* into *plane subgraphs*? The question points to the notion of *geometric graph thickness* or *drawing thickness* (defined in [3]), which differs from the classic thickness questions (graph thickness [39], book thickness [5], geometrical thickness [14]) which concern an inherent to the abstract graph attribute. In fact, this optimization problem is hard even on convex geometric graphs [10].

- Worst case, how many layers suffice for making possible that any geometric graph can be decomposed in that many plane subgraphs? This was the main concern of [3], accompanied with a conjecture for the answer.

- What is the *maximum cardinality* of a subset of the edges of the geometrical graph such that it consists of *pairwise non-crossing edges*? Best case, a triangulation appears as a subset of the edges.

The latest settled to be the main topic of this Thesis; apart from trying to reveal a camouflaged tiling, it can be seen as a greedy criterion for plane decomposition, though an approximation guarantee w.r.t. the optimal solution must be determined by further exploration and work on the problem.

1.2 Preliminaries

Throughout this Thesis, we focus on graphs drawn (embedded) with straight lines on $\mathbb{R}^2$. In order to simplify the language and the problems' formulation, we make extensive use of the terms *"geometric graph"* and (less often) *"graph drawing"*.

1.2.1 Geometric graphs and (fixed) graphs' drawings

Definition 1.1 (Geometric graph, ([6])). Given a set of points V in general position on $\mathbb{R}^2$ (i.e. no three are collinear) and a set E of straight line segments with endpoints in V, we call (V, E) a *geometric graph*. We may associate this *straight-line drawing* with the underlying abstract graph $G(V, E)$.

Since we have established a first association of a geometric object to an underlying abstract graph, it is extremely useful to technically "move" in the opposite way, in the sense of fixing a *drawing* of an abstract graph:

Definition 1.2 (Drawing of a graph G, ([3])). Given an (abstract) graph $G(V, E)$, a *(fixed) drawing* $D(G)$ on the plane consists essentially of a mapping function $D_V : V \to \mathbb{R}^2$. Then, for every $v_i v_j \in E$, the image $D(v_i v_j)$ is the line segment $D_V(v_i)D_V(v_j)$.

The above validate the use of the term "drawing of G" as an equivalent to "geometric graph" as soon as it is context-clear: if V and E were initially supposed to be points and segments on the plane, we use the term "geometric graph (V, E)"; if they are vertices and edges of an abstract graph, we make use of the term "drawing) $D(G)$". This pair of definitions seems useful whenever one may work in between graph drawing and combinatorial geometry.

1.2.2 Intersection vs. crossing of segments

As we are dealing with geometric graphs and especially their edges/segments, it is of importance to clarify once and for all the terms "intersecting" and "crossing" when referring to a pair of the above:

Definition 1.3 (Intersecting and crossing segments). Given two segments on $\mathbb{R}^2$, we will say that they *intersect* if they have at least a point in common; if this point is unique and not an endpoint of neither of the segments, then the two segments *cross*.

It will be mentioned, when necessary, that significant to our work graph classes are equal and computationally equivalent when defining them with either term (see [3], Chapter 5).

1.2.3 Convex geometric graphs

An important family of any geometric object is their confinement to be convex. The very same stands for our object of interest:

Definition 1.4 (Convex geometric graphs)**.** A geometric graph (V, E) is *convex* if all points V lay on the convex hull of V. Analogously to Definition 1.2, we define a graph's convex drawing $C(G)$, with the appropriate additional constraint that the image of V through C_V should be a convex set.

However, taking advantage of the observations and proofs of [3], Chapter 3, we may simplify the graph drawing definition:

Definition 1.5 (Convex graph drawing)**.** A convex drawing C of (the abstract) $G(V, E)$ on $\mathbb{R}^2$ is sufficiently defined by a mapping function $C_V : V \to [0..n-1]$.

This definition is equivalent to the σ-cycle defined by Bernhart and Kainen [5], as a relevant to their *book thickness* notion. It also appears in [10], where the problem of determining the σ-thickness of a graph is proved to be **NP**-hard. For us, it is useful to provide a few more definitions, in order to maintain a more elegant phrasing later on.

Definition 1.6 (Proper vertex labeling)**.** Given a convex geometrical graph, we will say that its vertex labeling $v_0, ..., v_{n-1}$ is *proper* if its n vertices appear in order, clockwise or counter-clockwise around its convex hull. Thus, we will assume that once we draw a graph via the convex mapping function C_V, we **relabel** to a proper labeling.

For our convenience, we will use modulo arithmetic for referring to the indexes of the vertices of a convex geometric graph, i.e. we shall admit that all edges $v_i v_{i+2}$, $i \in [0..n-1]$ are properly defined, as $v_x \equiv v_{x \bmod n}$, for any x, n. The following definitions may already validate the reasoning behind this choice.

In Chapters 3 and 4, we will extensively refer to certain edges of our convex graphs, called *span-2* edges. Consequelnty, let us define the edge span:

Definition 1.7 (Edge (or diagonal) span for a convex polygon)**.** Let V be a convex point set defining (among others) a convex n-gon on $\mathbb{R}^2$, and assume *proper vertex labeling*. For every edge (diagonal) $e = v_i v_j \in E$, we define its *span*, denoted by $|e|$, to be the minimum distance of the vertices it touches around the convex hull. For any edge $v_i v_j$ it is $|v_i v_j| = \min\{i - j \bmod n, j - i \bmod n\} \overset{\text{Def. 1.6}}{\equiv} \min\{i - j, j - i\}$.

1.2.4 Isomorphism in the context of convex geometric graphs

For abstract graphs, isomorphism is none other than the existence of a permutation p of the vertices between G and H such that vu is an edge of G *if and only if* $p(u)p(v)$ is an edge of H. We will need a stronger condition when working on convex geometric graphs; as an illustration of why this is critical, see Figure 1.1: graphs G and H are isomorphic, yet drawn via the same C_V their properties differ a lot.
As the key notion that can be properly defined in graph drawing, while the abstract graph lacks it, is *the notion of intersecting (and crossing) edges*, isomorphism should be defined with respect to this concept.

Definition 1.8 (Convex drawing isomorphism)**.** Two convex geometric graphs are isomorphic if the vertex relabeling preserves all intersections.

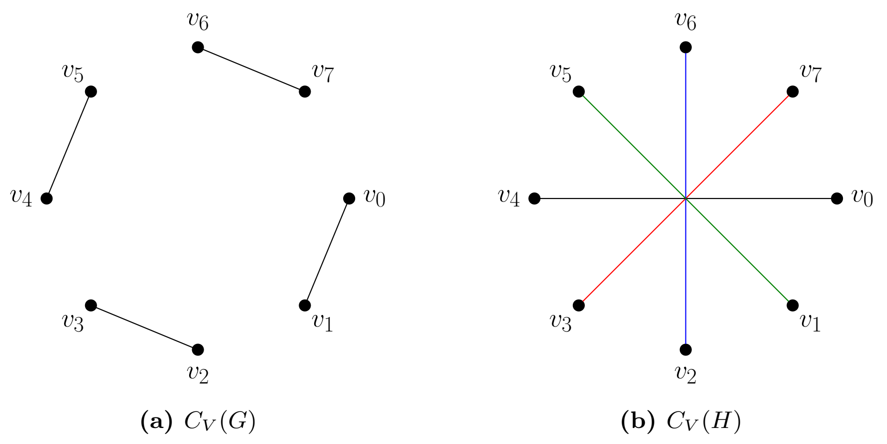


Figure 1.1: $G \simeq H$ but $C_V(G) \not\simeq C_V(H)$, as intersections are not preserved.

Proposition 1.9. For a given convex geometric graph (properly labeled), define a vertex r-rotation to be the permutation $p_r = [v_r, v_{r+1}, ..., v_{n-1}, v_0, ..., v_{r-1}]$ and a vertex mirroring to be the permutation $p_m = [v_{n-1}, v_{n-2}, ..., v_1, v_0]$. Applying the synthesis of any number of r-rotations or mirrorings, the image is a properly labeled graph, isomorphic to the initial. In a few words, **convex geometric graph isomorphism is closed under rotation and mirroring**.

1.2.5 Planarity

Central to graph theory is the notion of the planar graph.

Definition 1.10 (Planar graph)**.** A graph is called planar if it can be drawn on the plane without crossing edges. By Fáry's theorem [17], for any planar graph there is a straight line drawing certificate.

Within the context of geometric graphs, planarity loses its strength, as we are not concerned whether the underlying abstract graph is planar, we focus in the particular drawing on the plane (see Figure 1.2). As an analog, we will give the following definition:

Definition 1.11 (Plane (sub)graph)**.** Given a geometric graph (V, E), a (sub)graph (V', E') with $V' \subseteq V, E' \subseteq \binom{V'}{2}$ is characterized as *plane* if E' is crossing-free.

As an example, the subgraph $(V, \{v_0v_1, v_1v_2, v_0v_3, v_0v_2\})$ of the geometric graph of Figure 1.2b is one of its plane subgraphs.

Definition 1.12 (Outerplanar graph)**.** An abstract graph G is called *outerplanar* if it can be drawn on the plane without crossing edges and with all vertices on the outer region of the drawing ([8]); equivalently, no vertex is totally surrounded by edges.

G is outerplanar *if and only if* it does not contain $K_{2,3}$ and K_4 as minors. Equivalently, *if and only if* $G + K_1$ is planar. An outeplanar graph is maximal if no edge can be added to the drawing without losing outerplanarity. A maximal outerplanar graph $G(V, E), |V| = n$ may be embedded as a polygon triangulation. That gives us $|E| = 2n - 3$, the sum of the boundary edges, n, plus the number of diagonals in a triangulation, $n - 3$ ([20], [13]).

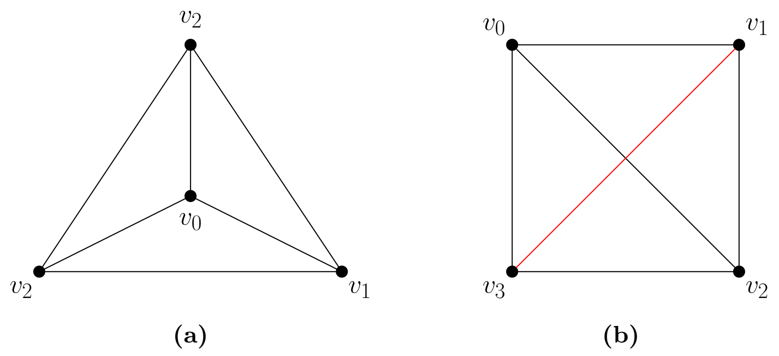


Figure 1.2: Two different drawings of K_4, the second not being a certificate of its planarity.

Simple polygons defined within our context

The given definitions allow us to clarify another few useful notions for this Thesis.

Definition 1.13 (Simple polygon and diagonals)**.** Given a geometric graph (V, E) with $|V| = n$, a simple polygon is a plane C_n subgraph of (V, E). Given a simple polygon, a diagonal is an edge lying entirely on its the inner region.

2 Triangulation existence problems on $\mathbb{R}^2$

A long existing result by Lloyd [26], dating back in 1977, states that: *"given a set V of points on the plane and a subset E of segments with endpoints in V, it is **NP**-hard to determine if a* triangulation *of V exists".* Even without a strict definition of triangulation, there should be only little confusion regarding what to search for in order to verify a "yes" instance. Also, it is easily noticed that the input of this very combinatorial problem is a geometric graph of ours, and having noted so, let us properly define what a triangulation is and introduce our notation, beginning with Lloyd's problem.

2.1 Formulating `TRI` using our notation

Definition 2.1 (Triangulation of a geometric graph)**.** Given a set of points V and segments $E \subseteq \binom{V}{2}$ on the plane, we will say that the set $T \subseteq E$ is a *triangulation* if no segments of T pairwise cross and $|T| = 3n - h(V) - 3$, where $h(V)$ is the number of points on the convex hull of V. Such a set of this size is the maximum possible (see Lemma 2.3).

Definition 2.2 (Triangulation existence (`TRI`))**.** Given a geometric graph, `TRI` is the decision problem whether a triangulation $T \subseteq E$ exists.

Lloyd [26] used a reduction from `SAT` to prove `TRI` is **NP**-hard. **NP** membership can be easily established, as verifying whether a guessed set T, $|T| = 3n - h(V) - 3$ does not include crossing segments (edges) can be clearly done in $\mathcal{O}(|T|^2)$ time. In the following sections we define and give the complexity of two similar problems which –in our opinion– derive quite naturally as we move along the points' generic-to-convex-position spectrum.

As a note: there is no doubt that the term "triangulation" is frequently used in a number of different contexts; so, we chose the term *"triangulation existence"* to attach to all three problems, a term which captures the output's inherence to the given input. Also, for sake of completeness, let us prove the simple lemma which links our triangulation to the maximum set of pairwise non-crossing segments on n points on the plane.

Lemma 2.3. Let T be a maximum set of pairwise non-crossing segments that can be defined on point set V with $|V| = n$ and convex hull $H(V)$ with size $|H(V)| = h(V)$. Then $|T| = 3n - h(V) - 3$.

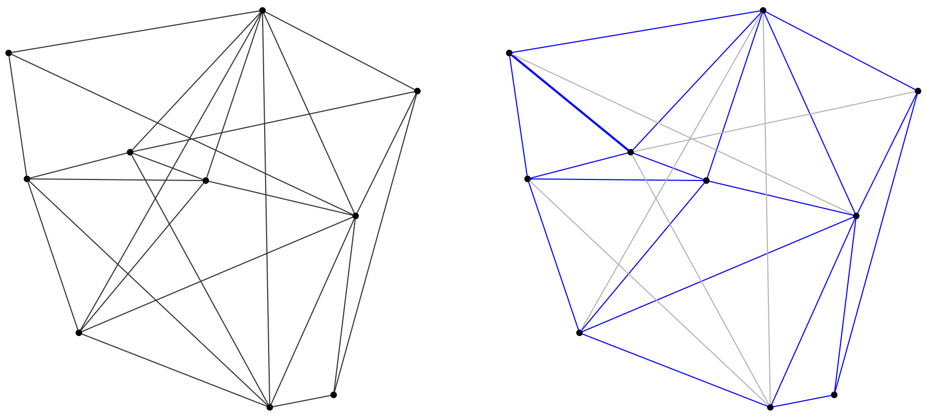

(a) A “no” instance of `TRI` **(b)** A “yes” instance of `TRI`

Figure 2.1: Triangulation existence given a geometric graph

Proof. It is easy to observe that whatever the set T, the subgraph (V, T) is a planar graph, as the drawing itself is a certificate for planarity. We now compare (V, T) to a maximal planar graph on n vertices, which has exactly $3n - 6$ edges and all its faces, including the outer face, bounded by 3 edges. In our embedding, which consists only of straight lines, we may well triangulate all faces but the outer face, which is necessarily bounded by $h(V)$ edges, therefore $h(V) - 3$ edges are missing from what would be a maximal planar graph: label the $h(V)$ points, $p_1, ..., p_{h(V)}$, (counter)clockwise around the convex hull and draw the non-crossing arcs $p_1p_3, p_1p_4, ..., p_1p_{h(V)-1}$. As a consequence, a triangulation T has size of exactly $3n - 6 - (h(V) - 3) = 3n - h(V) - 3$.

□

As reminder for Karp reductions

The most useful tool in order to prove **NP**-hardness (or **P** membership) of a problem is the *polynomial-time Karp reduction.* Maintaining the notation and approach of [4] we define:

Definition 2.4 ((Polynomial-time) Karp reduction)**.** We will say that a language L is Karp-reducible to L' if there is a polynomial time computable function $f : \{0,1\}^* \to \{0,1\}^*$, such that for every x it holds $x \in L \Leftrightarrow x \in L'$. We denote by $L \leq_p L'$.

In fact, let A be a problem defining language L^A of accepting inputs. If for some problem B it is shown that $A \leq_p B$, then B cannot be easier than A to solve, because then we would efficiently transform any instance x of A (string) via f and use the algorithm for B to decide if $x \in L^A$. The following are equivalent, given $A \leq_p B$:

- If A is **NP**-hard then B is **NP**-hard.
- If $B \in \mathbf{P}$ then $A \in \mathbf{P}$.

In Chapter 3 we will extend the notion of Karp reductions, to ensure similar deductions can be made w.r.t. counting the problems’ solutions. When referring to Karp reductions, we may omit to write it is polynomial time, as this prerequisite has become absolutely clear over time.

2.2 The convex point set case

It is quite without question that if a problem is defined on an arbitrary point set on $\mathbb{R}^2$, then its restriction to a convex point set will also be studied. No more to say, `Conv-TRI` is defined as follows:

Definition 2.5 (`Conv-TRI`)**.** Given a convex set of points V and segments $E \subseteq \binom{V}{2}$ on the plane, `Conv-TRI` is the decision problem whether a triangulation $T_C \subseteq E$ exists.

Here, having all points on the convex hull ($h(V) = n$) gives that the size of the triangulation T_C is equal to the size of a convex point set triangulation $3n - h(V) - 3 = 2n - 3$, which, of course, equals the size of the polygon triangulation.

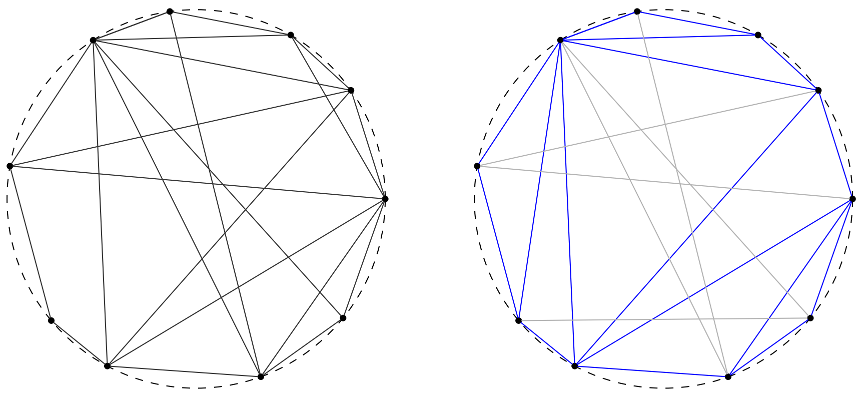

(a) A "no" instance of `Conv-TRI` **(b)** A "yes" instance of `Conv-TRI`

Figure 2.2: Triangulation existence given a convex geometric graph. Due to the properties of convex graph drawings ([3]), we are allowed to show any convex point set as on a single circle without loss of generality.

`Conv-TRI` is polynomially solvable

To easily prove the above, we need to focus on a specific graph class:

Definition 2.6 (CIRCLE graphs)**.** A graph is a CIRCLE graph if it is the intersection (or crossing) graph of chords in a circle.

The "crossing graph" is defined analogously to the intersection graph and both are essentially the following scheme: a vertex is created for every edge (or, generally, our objects that may conflict) and an edge is placed between two vertices if the respective objects they represent, i.e. the edges of the initial graph, *cross* (or, generally, conflict). Such a construction acts like a bridge between graph drawing and classic graph theory, as it attempts to capture the structure of a geometric object within a resulting abstract graph.

In [3], Chapter 5, we have shown that for CIRCLE graphs, altering the original definition to use the term "crossing graph" instead of "intersection graph" does not have an impact on the class itself[1].

Now we may present the theorem which relates our problem to the `Independent Set` on CIRCLE graphs. In a few words, taking advantage of the CIRCLE class membership, `Circle Independent Set` (`CIS`) can be seen as seeking a set of k pairwise non-crossing chords of a circle. The problem is known to be efficiently solvable (Gavril [19]) and most recent algorithms have improved the running time to $\mathcal{O}(n\min(d,\alpha))$-time, d being the density of the graph and α being its independence number (Nash and Gregg [29]). Note that both algorithms *maintain the independent set*, which can then be instantly accessed to work on.

Theorem 2.7. `Conv-TRI` *can be reduced in polynomial time to* `CIS`*.*

Proof. We need to show that any instance (V,E) of points and line segments on the plane, V being convex, can be transformed to an instance $V_c = f(V), E_c = f(E)$, where all $v \in V_c$ lie on a circle and for any two $e_1, e_2 \in E$, e_1 crosses e_2 if and only if $f(e_1)$ intersects $f(e_2)$.

All we need to do is to remind that placing the points of V around an arbitrary circle C in the same order of appearance as in V (function $f: V \to C$) is sufficient to guarantee that precisely every crossing of E is preserved by f and no new crossings appear (see [3], Chapter 3). This needs no more time than running a convex hull algorithm to find the order of the points of V ($\mathcal{O}(|V|\log h(V))$, [7]). To finalize the proof, we recall that $|T_C| = 2n-3$ (if such triangulation exists) and only need to run a `CIS` algorithm with input V_c, E_c and $k = 2n-3$. $\square$

2.3 Polygon triangulation existence (`Poly-TRI`)

A variation of the two aforementioned problems, that might be considered as an in-between problem, is to look for a *triangulated simple n-gon*,T_P, using the existent edges of a geometric graph. In order to be more clear, by the term *"simple"* we mean that the polygon does not cross itself, neither one may argue that it consists of more than n segments (see Figures 2.3a, 2.3c).

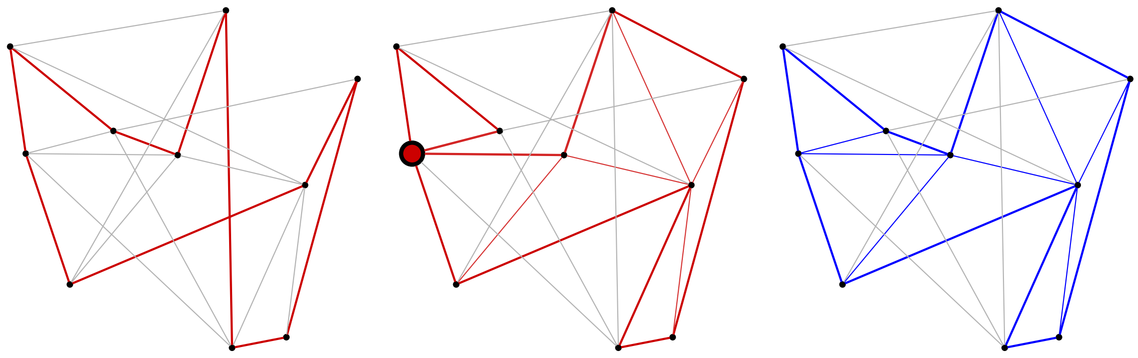
Note that there is not necessarily a single such polygon defined on a point set V, unless the point set is convex, in which case the problem coincides with `Conv-TRI`. Note also that:

- a triangulated n-gon will comprise exactly $2n-3$ segments ($|T_P| = 2n-3$);
- the abstract (V,T_P) will be a maximal outerplanar graph (see Definition 1.12);
- and, due to the above, (V,T_P) will have a Hamiltonian circuit.

Formally:

Definition 2.8 (`Poly-TRI`)**.** Given a set of points V and segments $E \subseteq \binom{V}{2}$ on the plane, `Poly-TRI` is the decision problem whether a set of $2n-3$ pairwise non-crossing segments $T_P \subseteq E$ exists, such that the subgraph (V,T_P) is maximal outerplanar.

[1]Same stands for SEG graphs, the intersection graphs of a set of segments on the plane.



(a) A "no" instance of Poly-TRI: all Hamiltonian circuits are degenerate polygons (one marked). **(b)** A "no" instance of Poly-TRI, with a marked degenerate n-gon (see the enlarged vertex). **(c)** A "yes" instance of Poly-TRI.

Figure 2.3: Polygon triangulation existence. The "no" instances also demonstrate *degenerate (non-simple)* polygons, which we do not search for.

Proving NP-complentess

Our definitions and observations lead to the following lemma:

Lemma 2.9. Poly-TRI $\in$ **NP**.

Proof. Guessing a set T_P of size $2n-3$, we may do the following (even using the most naive algorithms) in polynomial time:

1. check that no two edges cross ($\mathcal{O}(n^2)$);
2. check that every pair of adjacent vertices ($2n-3$ pairs) has at least one common neighbor ($\mathcal{O}(n)$ checks), thus belong to a triangle, and that for every such triangle there is no point of V lying inside (another $\mathcal{O}(n)$ checks).

□

Poly-TRI is NP-complete

A result of Wigderson (1982) states that *given a maximal planar graph, it is **NP**-complete to determine if it has a Hamiltonian circuit.* We will consider this (Max Planar HC) as our known **NP**-complete problem and establish a polynomial-time reduction to Poly-TRI. Eventually, we will need to efficiently draw any given instance G on the plane so that the occuring geometric graph (V, E) has a polygon triangulation *if and only if* the abstract G has a Hamiltonian circuit. Let us state:

Theorem 2.10. Max Planar HC *can be reduced in polynomial time to* Poly-TRI. *The reduction is parsimonious.*

Proof. Let G be our maximal planar graph. A straight-line drawing $D(G)$ can be constructed in $\mathcal{O}(n)$-time and onto a $(2n-4) \times (n-2)$ grid ([11], [9]).

Suppose that G has no Hamiltonian circuit. Then $D(G)$ cannot have a polygon triangulation T_P because this being true would force a Hamiltonian circuit in G; contradiction.

Now, suppose G has a Hamiltonian circuit C. Whatever the drawing D, $D(C)$ is a simple polygon on all n points/vertices and its interior region is triangulated because all faces of a maximal planar graph are triangles.

Ii is easy to demonstrate that the reduction preserves the number of solutions. Assuming two different Hamiltonian circuits, C_1 and C_2, of G and using the same argument as above, each cycle leads to obtaining a different polygon triangulation since the different sequences of the vertices around each circuit define different polygonal lines on the (drawn) graph. □

An illustrated example: we may consider the Golder-Harary graph, introduced in 1975 (Figure 2.4), the smallest of the class of maximal planar graphs which does not have a Hamiltonian circuit.

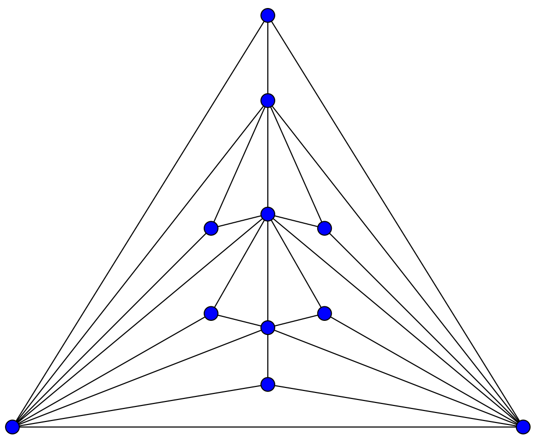


Figure 2.4: The Golder-Harary graph, $n = 11$, $m = 27$, drawn with straight lines.

2.4 Notes on triangulation existence

This subsection shall be considered an appendix, and it is presented here for any future reference.

Proposition 2.11 (Triangulation existence guarantee)**.** Let e_T^* be the minimum cardinality of E such that for **any** geometric graph (V, E), $|V| = n$ there exists a triangulation $T \subseteq E$. Then $e_T^* = \binom{n}{2}$.

Proof. Consider a complete geometric graph on point set V. If we remove just one of the segments joining two consecutive convex hull points, then there is no possible triangulation T. Thus, only a complete geometric graph guarantees a triangulation existence! Notice that the very same stands for the convex case. □

Proposition 2.12 (Polygon triangulation existence guarantee)**.** Let e_P^* be the minimum cardinality of E such that for **any** geometric graph (V, E), $|V| = n$ there exists a polygon triangulation $T_P \subseteq E$. Then $e_P^* \geq \frac{n^2-3n+4}{2} = \binom{n}{2} - (n-2)$.

Proof. Consider the complete graph on all points but $v_0 \in V$ (thus, there is a K_{n-1} in the drawing), plus an arbitrary edge incident to v_0, say $v_0v_x, x \in V \setminus \{v_0\}$. The graph has $\binom{n-1}{2} + 1 = \binom{n}{2} - (n-2)$ edges and does not allow any T_P, because v_0 does not belong in any triangle. $\square$

The conclusion drawn from the above is a simple one: bounds are not a useful tool to answering the questions we discussed.

Proposition 2.13. For every d there is a d-regular geometric graph G drawn on some set V, with $|V| = 2d^2 - 6d + 2$ such that there exists no triangulation $T \subseteq E$.

Proof. The construction is as shown in Figure 2.5. Each of the nodes in the rectangles form a complete bipartite graph $K_{d-3,d-3}$. The shaded regions indicate a $K_{1,d-3}$ star, while the red regions are empty of edges and *the witnesses for the lack of a triangulation.* $\square$

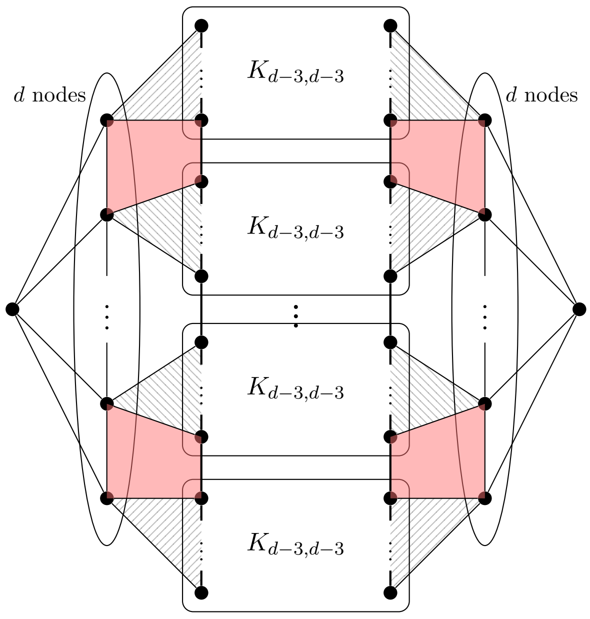


Figure 2.5: A d-regular geometric graph with no triangulation

3 Counting triangulations of geometric graphs

As described in the abstract, we shall bring Counting Complexity into play, in order to obtain new results and insights for our triangulation existence problems. First, we give a few words on the basics of Counting Complexity and describe the reductions used to show hardness results for **#P**. Then on, we deal with the counting versions of our problems.

3.1 Basics of Counting Complexity

The complexity class **#P** was introduced by Valiant [41], in an attempt to properly classify the problem of computing a 0-1 matrix's permanent.

Definition 3.1. **#P** is the class of functions that *count* the exact number of accepting paths of a polynomial-time nondeterministic Turing machine (PNTM).

Compared to **NP**, which determines *"only"* if there is at least one accepting path in the computation tree, one understands its power.

#P contains several interesting problems, which are actually the counting versions of classical **NP** problems. The class does feature *complete problems* e.g. #SAT, #CLIQUES. Note that for those two problems, their decision version is an **NP**-hard problem. However, there are also problems with easy decision version (in **P**) that are shown to be **#P**-complete under Cook reductions (#PERFECT MATCHINGS, #DNF-SAT).

Cook vs. Karp reductions. **#P** is not closed under Cook reductions (under reasonable assumptions). So, changing the reductions to Karp may yield a better characterization of counting problems with easy decision version (**#PE**). In this direction, the work of Pagourtzis and Zachos [30], along with the definition of the very interesting **TotP** (in [23]) and the results in [23] and [24], give a detailed overview of the structure and relations among counting complexity classes. Let us define:

Definition 3.2 (**FP**)**.** **FP** is the class of counting problems for which there exists a polynomial time deterministic Turing machine (PDTM).

Definition 3.3 (**TotP**, Kiayias et al. (1998))**.** **TotP** is the class of functions that count the total number of computational paths of a PNTM. In [30] it was shown that **TotP** is exactly the Karp closure of self-reducible functions of **#PE**. The problems corresponding to those functions are counting problems for which there exists a PNTM with exactly as many computation paths as the output value (plus 1).

Then, it is **FP** $\subseteq$ **TotP** $\subseteq$ **#PE** $\subseteq$ **#P**. Inclusions are proper unless **P=NP**. At the same time, **TotP** and **#P** are Cook-equivalent.

Reductions for counting problems

In Chapter 2 we have already made use of Karp reductions (defined in the conclusion in Section 2.1) for showing Theorems 2.7 and 2.10. Working in counting the solutions, a generic Karp reduction may not be sufficient to show that a difficulty in counting for A will translate to difficulty in counting for B; whereas, the following special case will do so (still following the main definition of [4]):

Definition 3.4 (Parsimonious Karp reduction)**.** We will say that a Karp reduction of A to B is *parsimonious* if the polynomially computable function f is a bijection of all witnesses for A to all witnesses for B. We will denote by $A \leq_p^{\text{1-1}} B$.

Definition 3.5 (Weakly parsimonious reduction)**.** We will say that a Karp reduction of A to B is *weakly parsimonious* if for the mapping function f, there is a polynomially computable function g such that A has $\#A$ witnesses *if and only if* B has $g(\#A)$ witnesses. We will denote by $A \leq_p^{\text{1-m}} B$.

The term "weakly parsimonious" often appears in bibliography, though there are a number of relevant terms, *one-many*, *many-one*, even the very special definition and case of *bit-shifting* reductions [25]. We choose to use the particular term as it feels natural, while we choose the notation commonly used to indicate the Karp *one-many* reductions, as they bear strong resemblance to the reductions presented in this chapter.

Above all, let $\#A$ be the problem of counting the witnesses of A, and $A \leq_p^{\text{1-m}} B$. Then, if $\#B$ was easier than $\#A$, we would transform any instance of A via f, solve $\#B$ as a subroutine and decide that $\#A = g^{-1}(\#B)$.

As a notation, we may well write $\#A \leq_p^{\text{1-m}} \#B$, as there is no confusion regarding the meaning.

3.2 Counting complexity for the convex case

Lemma 3.6. `#Conv-TRI` $\in$ **#PE**, as an immediate consequence of Theorem 2.7

In order to build the non-deterministic poly-time algorithm which proves **TotP** membership for `#Conv-TRI`, we need the following:

Proposition 3.7. Let T_C be a triangulation of a convex geometric graph. Then T_C has at least one edge of span 2.

Proof. Suppose this is not true, thus there is some triangulation $T'_C \subseteq E$ that does not include such an edge, in other words, the minimum edge span in T'_C is at least 3.

Therefore, given an edge of minimum span, say v_iv_{i+3} (the proof is similar whatever the minimum span considered), it is easy to see that the quadrilateral $v_iv_{i+1}v_{i+2}v_{i+3}$ has none of its diagonals in T'_C, otherwise the latter would have an span-2 edge (Figure 3.1 – in general, if k is supposed to be the minimum span, we obtain a $(k+1)$-gon with no diagonals). Together with the fact that T'_C includes by definition no pairwise crossing edges, we reach a contradiction: T'_C is not a triangulation, as it is not a maximal set of pairwise non-crossing edges. □

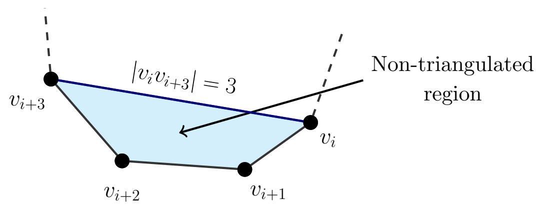


Figure 3.1: Illustrating the proof of Proposition 3.7

Proposition 3.7 is all we need to prove our self-reduction argument of this Chapter. However, there is a stronger one (Proposition 4.2), which we will present in Chapter 4. Let us add that we selected to mention both Propositions as it may contribute in a slightly better overall understanding.

Theorem 3.8. `#CONV-TRI` $\in$ ***TotP***

We establish the membership of the problem in **TotP** by proving the correctness of Algorithm 1, plus, of course, its polynomial running time. *For the rest of this section, we will make use of the traditional graph notation by a single letter (e.g. G)*, but always assume this graph is fixed on a convex position on $\mathbb{R}^2$.

The algorithm is based on some simple properties of our problem, which together add up to the desired result:

1. a trivial note, that the number of triangulations of a graph equals the number of triangulations which include a specific edge e plus the number of triangulations of $G \setminus e$.
2. Proposition 3.7, concerning the existence of span-2 edges in a triangulation,
3. and a self-reduction argument proved directly below.

Proposition 3.9 (Self-reduction argument for `#CONV-TRI`)**.** Let $G = (V, E)$ be a convex geometric graph and `#CONV-TRI`(G) the number of existing triangulations. Then, for any span-2 edge $e = v_{i-1}v_{i+1}$ we have that `#CONV-TRI`$(G \,|\, e \text{ included})$ = `#CONV-TRI`$(G \setminus v_i)$.

In other words, splitting our convex geometric graph w.r.t. a span-2 edge e has the property that the number of triangulations of G including edge e is equal to the number of triangulations of the induced convex graph on vertices $v_0, ..., v_{i-1}, v_{i+1}, ..., v_{n-1}$.

Proof. Let $v_{i-1}v_{i+1}$ be part of triangulations $T_C^1, ..., T_C^k$, $k \geq 0$ (notice that the proof also stands for the set of triangulations being empty). As a triangulation comprises no crossing pair of segments, then none of the above triangulations includes an edge incident to v_i, except for the mandatory for any triangulation $v_i v_{i-1}$ and $v_i v_{i+1}$ of the graph's perimeter. Therefore, any other edge of any T_C^i can be seen as an edge of the convex graph induced from G by deleting vertex v_i. This proves the self-reduction argument, as for any triangulation obtained for $G \setminus v_i$, we may obtain exactly one triangulation of G.

□

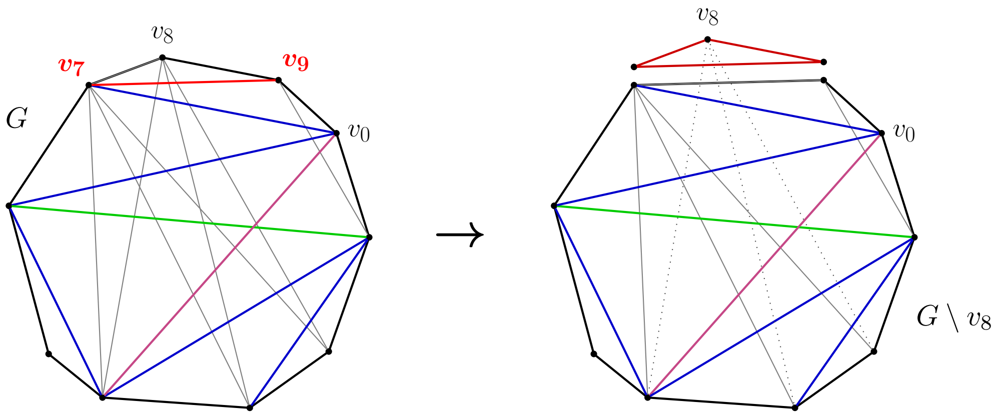


Figure 3.2: Splitting G w.r.t. edge v_7v_9. G has 2 triangulations featuring v_7v_9: the set of blue edges plus either the green or the magenta edge. The very same stands for $G \setminus v_8$.

Algorithm 1 runs in polynomial time. This is easily got by Lines 8 and 9, where it is evident that any child computation is performed on a smaller graph. Worst case, by Line 9, $\mathcal{O}(n^2)$ child computations are invoked, each running in poly-time.

3.3 Counting complexity of `Poly-TRI`

To the best of our knowledge, the most comprehensive chain of reductions, starting by a well-established **NP**-hard problem and leading to our `Poly-TRI`, is the following:

$$\begin{aligned} \texttt{3SAT} &\leq_p \textsc{3-connected Cubic Planar Hamiltonian Circuit} && [18] - \quad (3.1)\\ &\leq_p \textsc{Max Planar Hamiltonian Circuit} && [42] - \quad (3.2)\\ &\leq_p \textsc{Poly-TRI} && \text{Theorem 2.10} - \quad (3.3)\end{aligned}$$

Counting reductions. The technique we will use to prove **#P**-hardness results, while preserving the transitivity of the reductions, is to demonstrate that the above Karp reductions are *weakly parsimonious*.

(3.1) For the first reduction of the chain, [33] claims to have proven what we need; however, [25] argues that the proof is erroneous, while at the same time tweaks the original reduction by Garey et al. [18], in order to achieve one appropriate for demonstrating **#P**-hardness. For those purposes, the known **#P**-complete problem becomes #$\hat{3}$SAT (shown

Algorithm 1 Non-deterministic algorithm for #Conv-TRI

```
    Input: Convex geometric graph $G = (V, E)$
    Output: #Conv-TRI, the computation tree has exactly #Conv-TRI+1 leaves
 1: if Conv-TRI($G$) yields "NO" (equivalently #Conv-TRI($G$)=0) then
 2:     Stop
 3: else
 4:     Non-deterministically choose between {Stop, call GenTree($G$)} (*added dummy path*)
 5: end if
 6: procedure GenTree($G$)
 7:     Select any span-2 edge in the triangulation that forced us here, say $e = v_{i-1}v_{i+1}$
            (*it exists –Proposition 3.7– and can be found easily –properties of Conv-TRI/CIS algorithms*)
 8:     $\boldsymbol{G_0} := G \setminus \boldsymbol{v_i}$                    (*self-reduction argument*)
 9:     $\boldsymbol{G_1} := G \setminus \boldsymbol{e}$
10:     if Conv-TRI($G_0$)="YES" and Conv-TRI($G_1$)="YES" then
11:         Non-deterministically choose between {call GenTree($G_0$), call GenTree($G_1$)}
12:     else if Conv-TRI($G_0$) = "YES" then
13:         call GenTree($G_0$)
14:     else if Conv-TRI($G_1$) = "YES" then
15:         call GenTree($G_1$)
16:     else
17:         Stop
18:     end if
19: end procedure
```

by Valiant [40]) and the hardness is proved for the bigger class of *Cubic Planar* graphs. In all, we obtain that:

Theorem 3.10 (Liśkiewicz et al. (2003)). *The* Cubic Planar Hamiltonian Circuit *problem is #**P**-complete.*

Due to the technicality of the reduction, we omit sketching it, as it would occupy an unnecessarily large part of the Thesis, without offering any new perspectives or a better intuition. Let us note though, that while [18] talks about the 3-connectivity of the build by the reduction Cubic Planar Graphs, this is not a necessary condition for the reduction in [42] to hold true. In fact, Garey et al. probably spotted the property and wanted to indicate the difficulty in solving the HC problem within an even smaller graph class. Liśkiewicz et al. [25] fail to mention the 3-connectivity as a property of the constructed graphs, indeed.

(3.2) Focusing on the Karp reduction in [42] which proves 3-connected Cubic Planar Hamiltonian Circuit $\leq_p$ Max Planar Hamiltonian Circuit (3.2), we observe the following:

Corollary 3.11 (Observation on [42])**.** Let A be an instance of a *Cubic Planar* graph with n vertices. In *poly*-time we can construct a *Maximal Planar* graph B with $55n$ verites in such way that: A has k Hamiltonian circuits *if and only if* B has $k \cdot 64^n$ Hamiltonian circuits.

As a proof, we carefully go through the construction in [42]. Widgerson manages to meticulosly build a graph N (see Figure 3.3) with the following final properties and resulting lemma:

- N is a planar graph on 55 vertices;
- all its faces but the outer (which is a hexagon) are triangles, in other words N has $3 \cdot 55 - 6 - 3 = 156$ edges;
- there are exactly 64 Hamiltonian paths between any two w-labeled vertices.

Lemma 3.12 (Wigderson [42])**.** Let G be a graph which has N as an induced subgraph such that only its vertices of the outer face –labeled z or w– are adjacent to edges not in N. Then:

(a) In any Hamiltonian circuit of G, all vertices of N appear consecutively between two w-labeled vertices (marked as red, blue and green paths in Figure 3.3).

(b) Let e be an edge incident to a z-labeled vertex of N. If $e \notin N$ then it cannot participate in any Hamiltonian circuit of G.

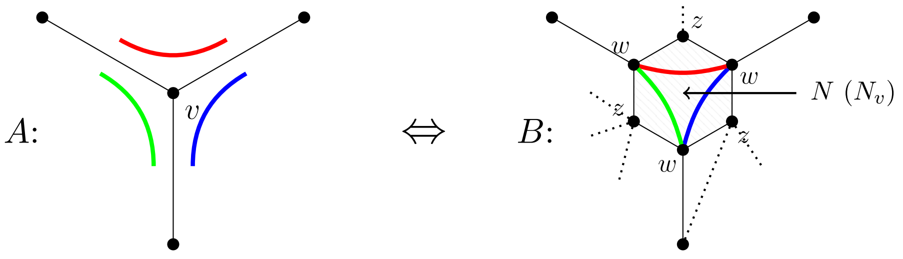


Figure 3.3: Core of the reduction by Wigderson [42], based on the “gadget” subgraph N

In a nutshell, the construction consists of replacing each vertex v of A (a 3-connected cubic planar graph) with a copy of graph N (denoted by N_v). Then, the resulting graph can be quite easily completed into a maximal planar graph B, with the desired property that A has a Hamiltonian circuit *if and only if* B has a Hamiltonian circuit. The idea is partially illustrated in Figure 3.3: for every $v \in A$, if a Hamiltonian circuit traverses it as the green (or blue or red) arc suggests, then for graph B there is a Hamiltonian circuit traversing subgraph N_v using the green (or blue or red) w-labeled pair as entry and exit vertices.

More importantly to us, Lemma 3.12(b) proves stronger than only guaranteeing the *if and only if* or the reduction: it guarantees the 1:64 ratio of traversing each vertex in A vs. each subgraph N in B as part of a Hamiltonian path. Essentially, the arguments of [42] suffice to prove this claim, which we state as Corollary 3.11.

Note. The *3-Connectivity* of the initial graph is not a prerequisite for the validity of the reduction in [42], as the construction is only based on the graph being *Cubic*.

(3.3) For the last reduction, we simply recall Theorem 2.10: it is parsimonious.

The overall result. Combining the above, we have indeed shown **#P**-hardness for `#Poly-TRI`, following the weakly parsimonious reductions 3.4 – 3.6.

$$
\begin{aligned}
\hat{3}\mathtt{SAT} &\leq_p \textsc{Cubic Planar Hamiltonian Circuit} && \text{[25] –} \qquad (3.4)\\
&\leq_p \textsc{Max Planar Hamiltonian Circuit} && \text{Corollary 3.11 –} \qquad (3.5)\\
&\leq_p \textsc{Poly-TRI} && \text{Theorem 2.10 –} \qquad (3.6)
\end{aligned}
$$

Theorem 3.13. `#Poly-TRI` *is* ***#P****-complete.*

4 An exact uniform sampling algorithm for convex triangulations

In this Chapter, we present an elegant algorithm for uniform sampling and coding of convex triangulations. The algorithm runs in a total $\mathcal{O}(n^2)$ time, outperforming any algorithm to our knowledge. Moreover, we may reasonably consider the first of the two distinct processes of the algorithm as the *preprocessing step*, running in $\mathcal{O}(n^2)$ time, as one may need to generate multiple triangulations of convex polygon(s) of the same size. This leaves us with another $\mathcal{O}(n^2)$ time random generation/coding step, which is again a significant improvement over any known algorithm. Interestingly, our method of analyzing convex triangulations yields a two-parameter recursion where Catalan numbers appear.

4.1 Related work

The consideration of different triangulations of a point set first appears in a well-studied form in the middle of the 18th century by Leonhard Euler: he successfully conjectured the closed formula for the *number of triangulations* of the *convex* n-gon, that was what we denote now by C_{n-2}, the $(n-2)$-th *Catalan number.* These numbers, named after the Belgian mathematician Eugène Charles Catalan, satisfy the following basic relations:

$$C_{n+1} = \sum_{k=0}^{n} C_k C_{n-k}, \quad C_0 = 1 \tag{4.1}$$

$$C_n = \frac{1}{n+1}\binom{2n}{n}, \quad C_0 = 1 \tag{4.2}$$

They occur as the solution of a very large number of counting problems in combinatorics. Stanley [38] gives more than 200 interpretations, more than 60 exercises, in all, a spectacular volume of work centered around the Catalan numbers.

While for convex graphs it is the Catalan numbers that give the exact number of triangulations, no such formula exists for the generic case. Also, enumerating the triangulations is not an easy task: already $C_n = \Theta(n^{-3/2}4^n)$, while the best known bounds for the generic case are currently set a lower $\Omega(2.43^n)$ (Sharir et al. [36]) and an upper $\mathcal{O}(30^n)$ (Sharir and Sheffer [34]). Naturally, there has been also work on counting the triangulations given a specific point set asymptotically faster than by enumerating all triangulations ([1]), and approximately counting with favorable compromises ([2]).

Another modern view of the same core problems is to efficiently generate random triangulations of point sets. There has been significant work on such problems, especially since the 1990s. Epstein and Sack [16] propose an $\mathcal{O}(n^4)$ algorithm for exact sampling over the triangulations of any simple polygon, which include the convex ones. This result appears to be improved by Ding et al. [15] to an $\mathcal{O}(n^2 + |E|^{1.5})$ time, where E is the set of diagonal edges (which lie inside the simple polygon). Moreover, the authors divide the algorithm in a preprocessing step, requiring $\mathcal{O}(n^2)$ time, and the main sampling procedure, requiring the $\mathcal{O}(|E|^{1.5})$ time. We note that for the convex n-gon, the maximum of $n(n-3)/2$ interior diagonal edges occurs, therefore the running time of the aforementioned algorithm is dominated by an $\mathcal{O}(n^3)$ term. The works of Denny and Sohler [12] and Poulalhon and Schaeffer [32] give efficient coding and sampling algorithms for plane triangulations seen as maximal planar graphs. Sharir and Welzl [35] also deal with maximal planar graphs with n interior points, but focus on obtaining bounds on the vertices' degree and use the result to get bounds on the total number of such graphs (see also [36, 34]).

An alternate way to attack triangulations is by means of structures and procedures which relate and move along the similar ones, most commonly triangulations that differ by a single edge flip: The work of Hurtado and Noy [22] builds a tree of convex triangulations, with a parent-child relationship fixed by an edge flip; Molloy et al. [28] and McShine and Tetali [27] build random walks on relevant structures using the same principle and study their mixing time – the latest obtaining an approximate sampling algorithm running in $\mathcal{O}(n^5)$ time. Parvez et al. [31] give an algorithm to generate all triangulations of a simple polygon, essentially in the same manner as in [22], in their attempt to triangulate a triconnected planar graph, introducing the measuring of time complexity per output triangulation.

4.2 Preliminaries

When counting, sampling, coding or generating convex triangulations, it is established that we assume a n-gon in convex position on $\mathbb{R}^2$, and all potential diagonals can be used to obtain a triangulation. Note the difference to our triangulation *existence* problems, where the diagonals can be selected only if they appear in the given edge set E. Thus, to take advantage and easily translate our aforementioned notation, algorithms, propositions and proofs in this context, we simulate our assumption by working within the subclass of *complete convex (geometric) graphs.* Briefly:

Remark 4.1. All triangulations of a convex n-gon are exactly all the triangulation instances T_C of the convex geometric K_n.

The convex K_n. We will often use the term *"convex K_n"* as an alternate to the complete convex graph, taking into account that all convex drawings of K_n are isomorphic, and therefore there is *a single* complete convex graph (convex K_n).

Consecutive vertices and span-2 edges. As an analog to the notion of consecutive vertices, given a proper labeling of the convex polygon, we define two *consecutive span-2 edges* to be a pair of edges of the form $v_{i-1}v_{i+1}$, v_iv_{i+2}. Moreover, we know that K_3 has no span-2 edges, K_4 has two and, for $n \geq 5$, K_n has exactly n of such edges. For $n \geq 5$ and any vertex v_i, we denote by e_i the span-2 edge $v_{i-1}v_{i+1}$, in other words, the edge that may form a triangle together with v_i.

The $\boldsymbol{K_{n,-m}}$ graph. We will need to work on specific graphs, that allow for the desired structure to be witnessed. We define $\boldsymbol{K_{n,-m}}$ to be the nearly complete convex K_n which **misses only $\boldsymbol{m}$ consecutive span-2 edges**. Therefore:

- $K_{n,0} \equiv K_n$ and when properly defined, K_{n-m} has $\binom{n}{2} - m$ edges.
- For $n \geq 5$, it is $-n \leq -m \leq 0$, as a graph may miss up to all n span-2 edges.
- The graphs $K_{4,-1}$, $K_{4,-2}$ are properly defined.
- For fixed n, m, all $K_{n,-m}$ are isomorphic to each other (under rotation, Proposition 1.9).
- Finally, we will mark by $T_{n,-m}$ the number of triangulations of $K_{n,-m}$. To practice on the notation, remember we know $T_{n,0}$ for all n, it is $T_{n,0} = C_{n-2}$.

Proposition 4.2. Every triangulation of a convex geometric graph on 5 or more vertices has at least two span-2 edges.

Proof. A triangulation T_C on n points requires that all faces but the outer are triangles. The number of faces f is equal to $n-1$ (e.g. use Euler's characteristic on plane graphs), therefore there is a total of $n-2$ inner faces/triangles. Since all n non-diagonal edges of T_C are sides of the $n-2$ triangles, by the pigeonhole principle, there are at least 2 triangles which use 2 sides of the n-gon as their sides. Then, those triangles' third edge must be an span-2 one and, as soon as $n \geq 5$, those third edges do not coincide. □

Proposition 4.3. A convex geometric graph with $n \geq 5$ and only two2 span-2 edges which appear *consecutive* in the drawing has no triangulation. Equivalently, $T_{n,-(n-2)} = 0$.

Proof. Two consecutive span-2 edges intersect, therefore together they do not form any triangulation, unless only one is needed (quadrilateral). So there must be a third span-2 edge which, together with one of the 2 consecutive ones, is the obligatory second span-2 edge in a triangulation T of the graph (see Proposition 4.2). Contradiction, as the graph has no other such edge. □

4.3 The reduction argument and recursive relation

In Chapter 3, we presented a self-reduction argument based on partitioning the number of existing convex triangulations T_C into those which include a specific span-2 edge e and those which do not (Proposition 3.9). Of course, if our input (unnecessarily) is the *complete convex graph*, the non-deterministic Algorithm 1 built around the self-reduction will give us the number of triangulations of the convex n-gon, i.e. the known C_{n-2}.

Our idea. A non-deterministic algorithm for counting in **TotP** can be transformed into an (approximate) sampling algorithm, as long as each subtree of a node of the computation tree is selected with probability (approximately) proportional to its size (see Sinclair and Jerrum [37]). Using Proposition 3.9, a convex K_n, *known to have* C_{n-2} *triangulations* is reduced to a convex K_{n-1}, *known to have* C_{n-3} *triangulations* and a $K_n \setminus e$, which evidently has the remaining $C_{n-2} - C_{n-3}$ triangulations (see Figure 4.1). However:

Remark 4.4. For the subclass of complete geometric graphs, Proposition 3.9 does not prove a self-reduction argument.

In other words, we may know how to branch for a first time, but step 7 of Algorithm 1 will then prove insufficient. Eventually, we will work on the basis of the same technique as in Chapter 3, this time with some added rules which reveal a hidden structure of the reduction for this new problem.

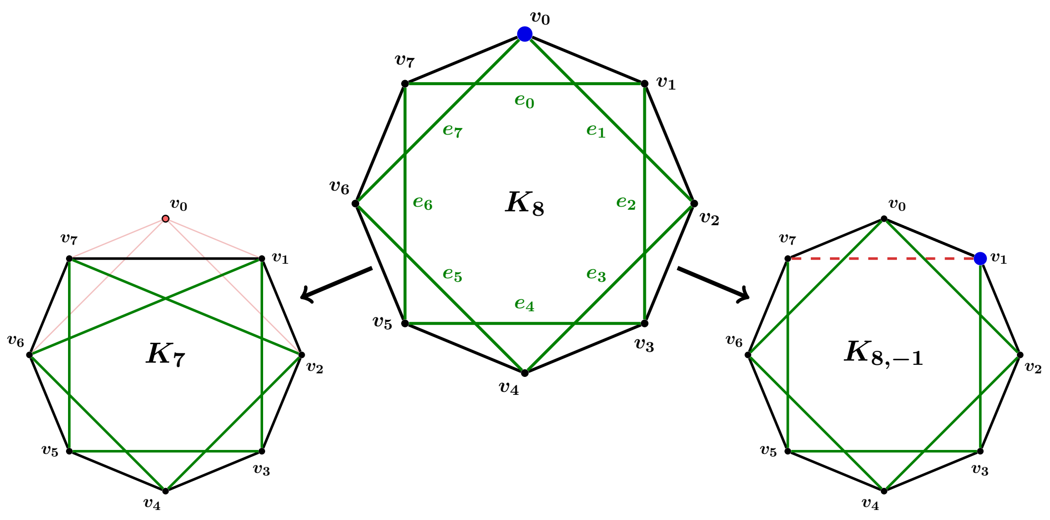


Figure 4.1: Reduction for the complete K_n, working on v_0/e_0

First relation. We have actually described the first of the equations defining the sought-for recursive relation (the proof is omitted), as for the complete K_n is reduced easily (again, see Figure 4.1):

$$T_{n,0} = T_{n-1,0} + T_{n,-1} \tag{4.3}$$

The left subtree will feature all triangulations of a K_7, which become all triangulations of the initial K_8 which include e_0; while the right subtree features all triangulations of the octagon which do not include e_0. Note that we can easily store this information for selecting each of the subtrees in a *single bit* of an adjacency matrix of the graph/polygon, that is the one indicating the existence of e_0. Of course, this will hold true for all branches of the tree, as they are all created the same way.

Second relation. Now, we must reduce any $K_{n,-m}$ in some orderly fashion. First, observe Figure 4.2 for the case of $m = 1$, especially the marked blue vertex v_1. If we work on the selection or not of e_1 for the next branching of our tree, we obtain either a $K_7 \equiv K_{7,0}$ (left child), either a $K_{8,-2}$. In general (Figure 4.3), a $K_{n,-m}$ is reduced to either a $K_{n-1,-m+1}$ (left child) or a $K_{n,-m-1}$ (right child).

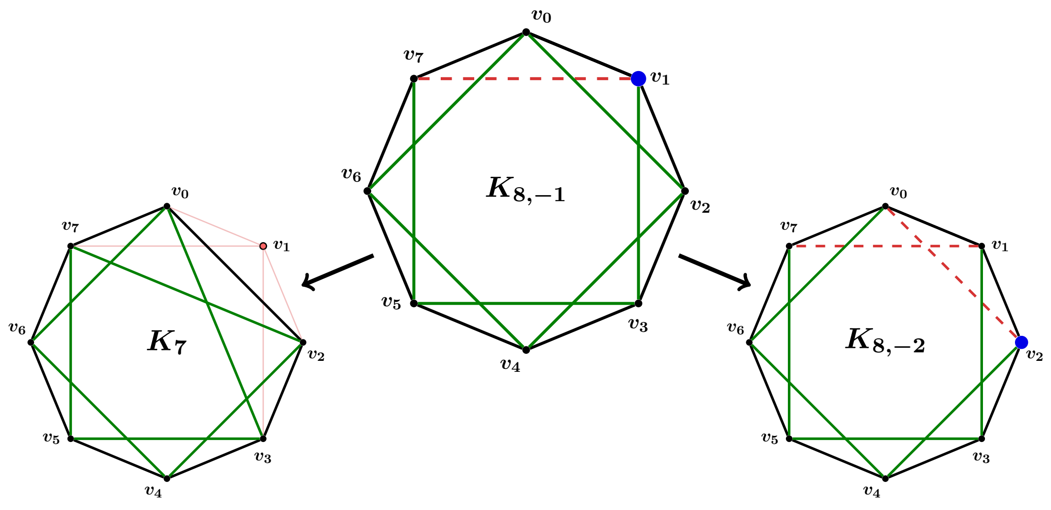


Figure 4.2: Reduction for $K_{n,-1}$, working on the *next* vertex/span-2 edge

The key. It proves that the key to our reduction is the correct definition of the **next vertex (or span-2 edge)** to work on. If, for instance, while on $K_{8,-1}$ we select to work on v_5/e_5, the nice to our reduction K_7 does not pop out. More importantly, we would miss the recurring of *isomorphic graphs*, which give us the number of triangulations they feature per class ($K_{n,-m}$), and not per specific instance.

Definition 4.5 (Next vertex/span-2 edge)**.** Given a convex geometric graph, **properly labeled***, missing only m consecutive span-2 edges, $e_i, e_{i+1}, ..., e_{i+m}$. The *next* vertex/span-2 edge is the pair v_{m+1}/e_{m+1}.

Now, let us formally prove our claim.

Theorem 4.6. *For any $K_{n,-m}$ and $-n+1 \leq -m$, it is*

$$T_{n,-m} = T_{n-1,-m+1} + T_{n,-m-1} \tag{4.4}$$

Proof. Without loss of generality (see 1), we may relabel our initial graph –if necessary– and have the edges $e_0, ..., e_{m-1}$ as the missing span-2 ones. The next vertex/span-2 edge is the pair v_m/e_m. The right subtree is rooted to a $K_{n,-m-1}$ graph, as it derives from deleting e_m, having a consecutive $m+1$ span-2 (but no other) edges missing.

For the left subtree, deleting v_m leaves us to examine the edges of a graph on $n-1$ vertices. We have that edges $v_{n-1}v_1, v_0v_2, ..., v_{m-3}v_{m-1}$ are all missing from the induced subgraph, and together they add up to $m-1$ consecutive span-2 edges. Edge $v_{m-2}v_m$, missing from $K_{n,-m}$ is not a potential edge of the smaller graph. Edge $v_{m-1}v_{m+1}$ has now become a side of the $(n-1)$-gon, and $v_{m-2}v_{m+1}$ and $v_{m-1}v_{m+2}$ are now span-2 edges of the new graph, as are all the (potentially) remaining $v_{m+1}v_{m+3}, ..., v_{n-2}v_0$. Thus, we have obtained a $K_{n-1,-m+1}$ graph. $\square$

As a comment, since the argument holds as long as e_m is present, and it can be $v_{m+1} \equiv v_0$, we get the $-n+1 \leq -m$ as a requirement; however, for our overall goal, proving the argument for down to $m = n-3$ would suffice, as we have shown that the right child of a $K_{n,-n+3}$ graph, that is a $K_{n,-n+2}$, has no triangulations.

The "properly labeled" part of Definition 4.5. The left child of a $K_{n,-m}$ in our reduction is a graph on $n-1$ vertices, in other words, w.r.t. our initial proper vertex labeling, the labeling of the left child is not proper, as a vertex is missing. Therefore, we must **relabel** this graph, and every left child graph of the tree. The following Rule gives the final ingredient for building the desirable reduction tree.

Rule 1. Let $K_{r,-m}$ be the left child of a node of a tree rooted at K_n. If $m = 0$, then relabel the complete K_r as desired. Else, place r_0 opposite to the first missing span-2 edge and complete with a proper relabeling. For each new graph, maintain a $1 \times n$ table to indicate the mapping of the current to the initial labeling; current missing vertices of the parent graph can be marked (e.g. -1 entries).

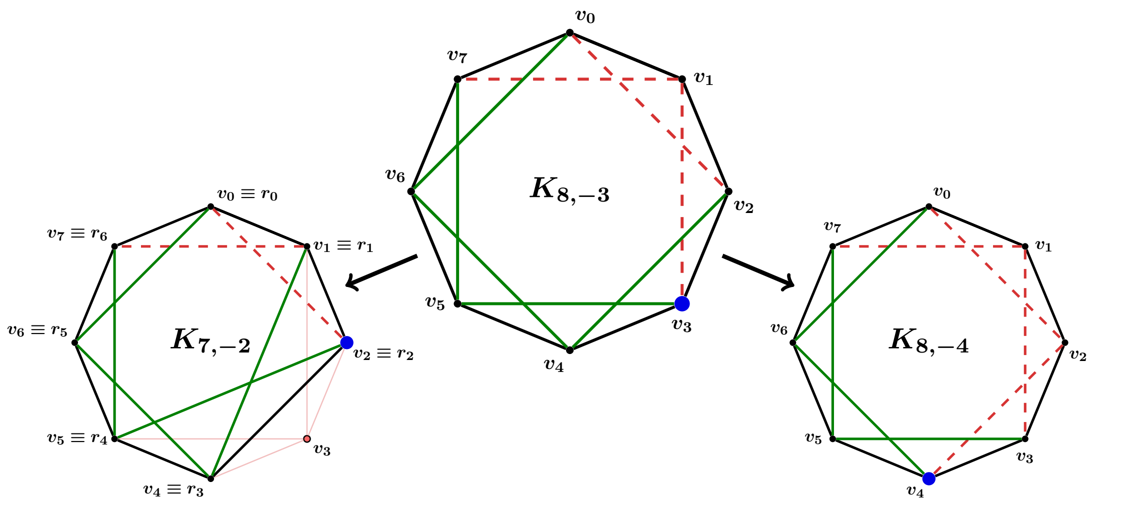


Figure 4.3: Reduction for $K_{n,-m}$

Defining the base cases. We are left to define the base cases of the recursive relations 4.3 and 4.4.

- Proposition 4.3 gives us $T_{n,-n+2} = 0$ for pentagons and up; but $T_{4,-2}$ gives the number of triangulations of a quadrilateral missing all 2 span-2 edges, therefore the relation is satisfied for all $n \geq 4$.
- $T_{3,0} = 1$, as K_3 is a triangle.

- $T_{3,-1}$ is not properly defined and not actually needed, but for sake of completeness, we shall consider it equal to zero.

Properties of the reduction tree

In all, we have build a triangulation tree with the following properties:

- Every node has 2 children (left and right – binary tree).
- Every node indicating $K_{r,-m}$ needs $\mathcal{O}(n)$ time to be created and coded: store integers r and m, the $1 \times n$ mapping matrix for the vertex labeling, plus a 2 integer indicator for backtracking to the parent node: if it is a left child, hold $[v_m, -1]$, for a right child hold $[v_{m-1}, v_{m+1}]$ where v_m is the critical vertex of the parent node w.r.t. which the node branched (store as labeled in the parent node).
- For any node, the triangulations coded in the left subtree are disjoint to the ones of the right subtree, as they differ at the edge w.r.t. which the node branched.
- If a node is $\boldsymbol{K_3}$, then stop and mark **one triangulation** (green leaf). Due to the above, all green leaves are left children and there is a total of exactly $\boldsymbol{C_{n-2}}$ green leafs in the tree.
- If a node is of the form $\boldsymbol{K_{r,-r+2}}$, then stop and mark **no triangulation** (red leaf)
- Considering all the above, from any green leaf, backtracking to the root is equivalent to obtaining one specific triangulation; this is achieved in $\mathcal{O}(n)$ time.

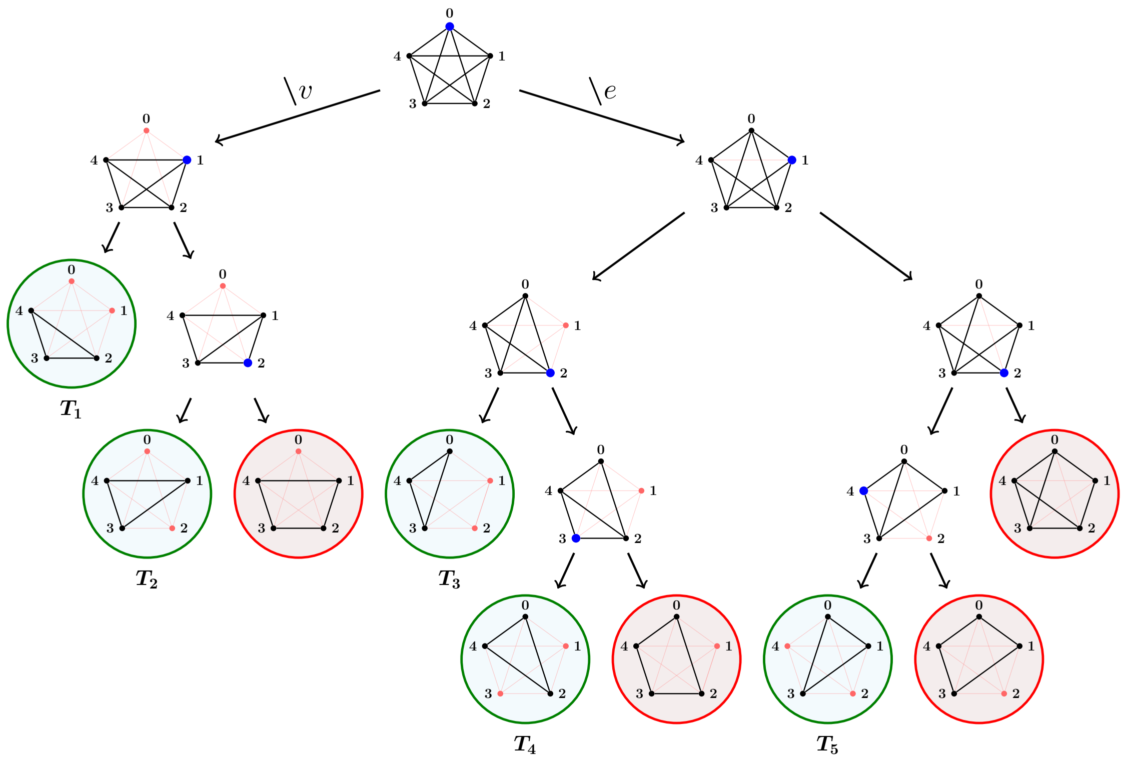


Figure 4.4: Full reduction tree for the convex pentagon. Blue color indicates the working on node. Relabelings are not marked. Each of its 5 triangulations is encoded in a different green leaf.

Figure 4.4 shows the full reduction tree for the convex K_5. To give a for instance, consider the green leaf encoding triangulation T_3. T_3 includes the triangle $v_0v_3v_4$ (end of reduction); as a left child of its parent node, which branched w.r.t. v_2/v_0v_3, it includes v_0v_3 as an span-2 edge of the parent K_4 subgraph, so triangle $v_0v_2v_3$ is in T_4. The K_4 parent node is a left child of the $K_{5,-1}$ node which branched w.r.t. v_1, therefore the triangulation does also include triangle $v_0v_1v_2$; finally (though not necessary), the $K_{5,-1}$ node being a right child of the K_5 which branched w.r.t. v_0 suggests that v_4v_1 is missing from the T_4.

Note that this is a small example, and some leafs (T_2, T_4, T_5) already indicate unambiguously the encoded triangulation. Observe also that even if it is not directly visible that T_1 differs from T_4, our simple branching rule guarantees that T_1 includes v_4v_1 as an edge, while T_4 does not, as they belong to a different subtree of the root node which branched w.r.t. v_0/v_4v_1.

4.4 The algorithm

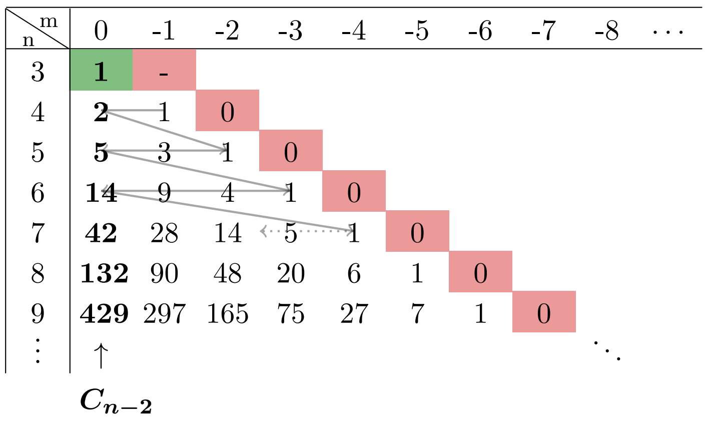
By now, getting the final algorithm for exact sampling over convex triangulations, should be almost straight-forward.

Exponential number of nodes, polynomial number of distinct isomorphisms. The key to our efficient algorithm is the structure we revealed, a structure that allows for the partitioning of an exponential number of nodes (only the green leaves give $\mathcal{O}(4^n)$ nodes) into polynomial number of classes. In fact, there are exactly $(n-3)(n-1)+1$ classes, within each, all nodes/graphs are isomorphic to a specific $K_{n,-m}$ characterized by a uniquely defined $T_{n,-m}$.

Calculating all $T_{n,-m}$ in $\mathcal{O}(n^2)$ time. At each node of the reduction (and computation) tree, a $T_{n,-m}$ quantity is attached, indicating the number of triangulations "hanging" from the subtree rooted at this very node. Due to the recursion we built (4.3, 4.4 and base cases), we are definitely able to calculate all these quantities. However, it is a lot easier to calculate all of them in advance, **bottom-up**, in $\mathcal{O}(n^2)$ time, i.e. $\mathcal{O}(1)$ time per single quantity. Figure 4.5 illustrates the table created by the pseudocode:

```
• Initialize $T_{3,0} := 1$ (green cell in Figure 4.5).
• For $3 \le k \le n$ initialize $T_{i,-i+2} := 0$ (red cells in Figure 4.5).
• For $i = 4$ up to $n$ do:
    1. For $j = i-3$ down to 1 do:
           $T_{i,-j} := T_{i-1,-j+1} + T_{i,-j-1}$
    2. $T_{i,0} = T_{i-1,0} + T_{i,-1}$
```

The filling of the table may well be considered as a **pre-processing step** for the algorithm, as only n determines its entries, and one may need a lot of triangulations of the same graph. Going even further, nothing deprives one of having a very large $T_{n,-m}$ table (its size in bits is $\mathcal{O}(n^3)$) and recall instantly whatever number they wish.



n \ m	0	-1	-2	-3	-4	-5	-6	-7	-8	⋯
3	**1**	-								
4	**2**	1	0							
5	**5**	3	1	0						
6	**14**	9	4	1	0					
7	**42**	28	14	5	1	0				
8	**132**	90	48	20	6	1	0			
9	**429**	297	165	75	27	7	1	0		
⋮	↑								⋱	
	$\boldsymbol{C_{n-2}}$									

Figure 4.5: The table of triangulations $T_{n,-m}$ for each $K_{n,-m}$. It has $(n-3)(n-1)+1$ entries. The arrows show the sequence in which the cells fill up.

Optimal coding and exact sampling

We know that $T_n = C_{n-2}$. The asymptotic behavior of Catalan numbers is the following:

$$C_n \leq \frac{4^n}{n^{3/2}\sqrt{\pi}} \qquad \text{and} \qquad \lim_{n\to\infty} \frac{4^n}{C_n n^{3/2}\sqrt{\pi}} = 1$$

Therefore, it is evident that $\mathcal{O}(n)$ bits suffice to encode all triangulations of the convex n-gon. Say we code all triangulations by exactly C_{n-2} consecutive integers $[0..C_{n-2}-1]$, in other words, *optimally*. Denote by $T[0]$ a first triangulation encoded as 0, by $T[C_{n-2}-1]$ the corresponding last triangulation, encoded by integer $C_{n-2}-1$.

Decoding and sampling. The two procedures are actually the same, the only difference lies in whether a coin is tossed to get a code x. Regarding the coin toss for sampling, calculate C_{n-2} and toss the minimum number of coins needed (of course this number is linear to n) to form an integer $x \in [0, C_{n-2}-1]$. It can be $x \geq C_{n-2}$ and if so, then toss again. The procedure ends in an expected ≤ 2 rounds and gives a uniformly random integer $0 \leq x \leq C_{n-2}-1$. This is the code of the sampled triangulation, proceed to the main Algorithm 2.

Coding. We are given a triangulation T_C of a convex n-gon triangulation, and proceed almost identically as in Algorithm 2; only now the branching condition checks the edges of the polygon (see Algorithm 3).

The branching rule. For both algorithms, it is easy to show they branch *with probability analogous to the triangulations of each of the two subtrees rooted at the two children* (see Figure 4.6). Overall correctness is easy to prove, too.

Algorithm 2 Sampling/decoding of convex triangulations

```
Input: Convex geometric $K_n$, the $T_{n,-m}$ table, (int $x$)
Output: A (random) triangulation $T_C = T[x]$

if we want to sample then
    Generate random $x$
else
    $x$ is given and already indicates a triangulation (decoding mode)
end if
$a := 0, b := C_{n-2} - 1$
while $a \neq b$ do
    if $x \in [a, a + T_{(\text{left child})} - 1]$ then
        branch left
        $b := a + T_{(\text{left child})} - 1$
    else
        branch right
        $a := a + T_{(\text{left child})}$
    end if
end while
Backtrack to the root and built and output $T_C$
```

Algorithm 3 Coding of convex triangulations

```
Input: Convex geometric $K_n$, the $T_{n,-m}$ table, triangulation $T_C$
Output: A code $x$ for which it is $T_C = T[x]$

$a := 0, b := C_{n-2} - 1$
while $a \neq b$ do
    if Follow Rule 1 and Definition 4.5. If the *next span-2 edge* is in $T_C$ then
        branch left
        $b := a + T_{(\text{left child})} - 1$
    else
        branch right
        $a := a + T_{(\text{left child})}$
    end if
end while
Output $x$
```

Tree height and time complexity

It only remains to show that all the tree traversing algorithms described above are efficient. Actually, they are $\mathcal{O}(n^2)$. This is due to two final properties we note hereby:

1. As every $T_{n,-m}$ can be recalled in $\mathcal{O}(1)$ time, any subroutine aiming to lead from a node to either of its children needs $\mathcal{O}(n)$ time (see 4.3).

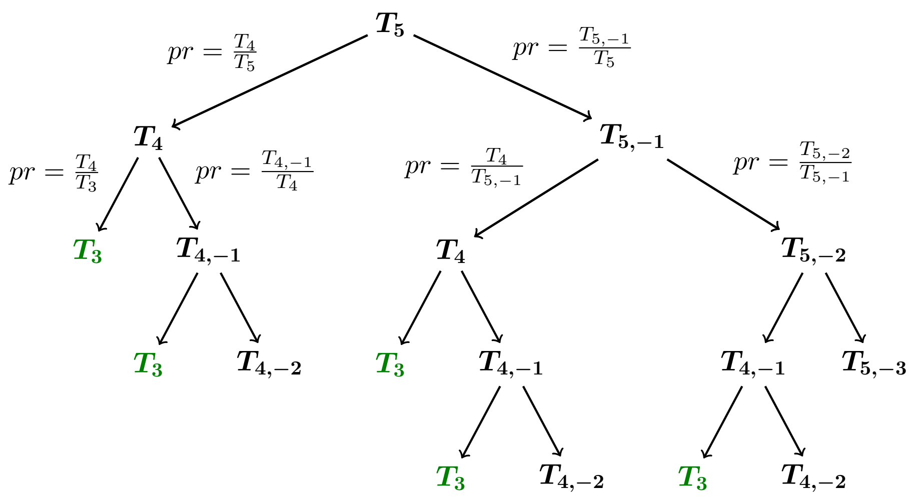


Figure 4.6: The universal sampling algorithm scheme: branching with probability analogous to the size of the subtree. The height of the tree is $\mathcal{O}(n)$.

2. The height of the tree is *linear*. A nice way to see that is with the help of the recursive relation and Figure 4.5: beginning at $T_{n,0}$, you may move 1 square to the north (left child) or 1 to the east (right child). This holds for anytime you land on the first column. Else, you may move either to the northwest square (left child) or the east square (right child). Reaching $T_{3,0}$ will need a maximum of $2(n-3) = \mathcal{O}(n)$ steps.

The above complete our analysis for the complexity and correctness of our algorithm. In all, there is an $\boldsymbol{\mathcal{O}(n^2)}$ time **exact sampling and optimal coding** algorithm, as well as a **decoding** algorithm for **convex triangulations** of a polygon on n vertices.

Bibliography

[1] V. Alvarez and R. Seidel. A simple aggregative algorithm for counting triangulations of planar point sets and related problems. In *Symposuim on Computational Geometry 2013, SoCG '13, Rio de Janeiro, Brazil, June 17-20, 2013*, pages 1–8, 2013.

[2] V. Alvarez, K. Bringmann, S. Ray, and R. Seidel. Counting triangulations and other crossing-free structures approximately. *Comput. Geom.*, 48(5):386–397.

[3] A. Angelopoulos. On the Variants of Thickness of a Graph, the Drawing Thickness of a Graph Drawing and Complexity. Diploma thesis, Dep. of CS, School of Electrical and Computer Engineering, National Technical University of Athens, 2012.

[4] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009. ISBN 0521424267, 9780521424264.

[5] F. Bernhart and P. C. Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, 1979.

[6] P. Bose, F. Hurtado, E. Rivera-Campo, and D. R. Wood. Partitions of complete geometric graphs into plane trees. *Comput. Geom.*, 34(2):116–125, 2006.

[7] T. M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 16(4):361–368, 1996.

[8] G. Chartrand and F. Harary. Planar permutation graphs. *Annales de l'institut Henri Poincaré (B) Probabilités et Statistiques*, 3(4):433–438, 1967.

[9] M. Chrobak and T. H. Payne. A linear-time algorithm for drawing a planar graph on a grid. *Inf. Process. Lett.*, 54(4):241–246, 1995.

[10] F. R. K. Chung, F. Thomson, Leighton, and A. L. Rosenberg. Embedding graphs in books: a layout problem with applications to VLSI design. *SIAM J. Algebraic Discrete Methods*, 8:33–58, 1987.

[11] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

[12] M. Denny and C. Sohler. Encoding a triangulation as a permutation of its point set. In *Proceedings of the 9th Canadian Conference on Computational Geometry, Kingston, Ontario, Canada, August 11-14, 1997*, 1997.

[13] S. L. Devadoss and J. O'Rourke. *Discrete and Computational Geometry*. Princeton University Press, 2011.

[14] M. B. Dillencourt, D. Eppstein, and D. S. Hirschberg. Geometric thickness of complete graphs. *J. Graph Algorithms Appl.*, 4(3):5–17, 2000.

[15] Q. Ding, J. Qian, W. Tsang, and C. Wang. Randomly generating triangulations of a simple polygon. In *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, Kunming, China, August 16-29, 2005, Proceedings*, pages 471–480, 2005.

[16] P. Epstein and J. Sack. Generating triangulations at random. *ACM Trans. Model. Comput. Simul.*, 4(3):267–278, 1994.

[17] I. Fáry. On straight line representation of planar graphs. *Acta Univ. Szeged. Sect. Sci. Math.*, 11:229–233, 1948.

[18] M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar hamiltonian circuit problem is np-complete. *SIAM J. Comput.*, 5(4):704–714, 1976.

[19] F. Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3:261–273, 1973.

[20] F. Harary. *Graph theory.* Addison-Wesley, 1969.

[21] J. E. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[22] F. Hurtado and M. Noy. Graph of triangulations of a convex polygon and tree of triangulations. *Comput. Geom.*, 13(3):179–188, 1999.

[23] A. Kiayias, A. Pagourtzis, K. Sharma, and S. Zachos. The complexity of determining the order of solutions. In *Proceedings of the First Southern Symposium on Computing*, 1998.

[24] A. Kiayias, A. Pagourtzis, and S. Zachos. Cook reductions blur structural differences between functional complexity classes. In *2nd Panhellenic Logic Symposium*, pages 132–137, 1999.

[25] M. Liśkiewicz, M. Ogihara, and S. Toda. The complexity of counting self-avoiding walks in subgraphs of two-dimensional grids and hypercubes. *Theoretical Computer Science*, 304(1):129–156, 2003.

[26] E. L. Lloyd. On triangulations of a set of points in the plane. In *FOCS*, pages 228–240, 1977.

[27] L. McShine and P. Tetali. On the mixing time of the triangulation walk and other catalan structures. In *Randomization Methods in Algorithm Design, Proceedings of a DIMACS Workshop, Princeton, New Jersey, USA, December 12-14, 1997*, pages 147–160, 1997.

[28] M. S. O. Molloy, B. A. Reed, and W. Steiger. On the mixing rate of the triangulation walk. In *Randomization Methods in Algorithm Design, Proceedings of a DIMACS Workshop, Princeton, New Jersey, USA, December 12-14, 1997*, pages 179–190, 1997.

[29] N. Nash and D. Gregg. An output sensitive algorithm for computing a maximum independent set of a circle graph. *Inf. Process. Lett.*, 110(16):630–634, 2010.

[30] A. Pagourtzis and S. Zachos. The complexity of counting functions with easy decision version. In *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings*, pages 741–752, 2006.

[31] M. T. Parvez, M. S. Rahman, and S. Nakano. Generating all triangulations of plane graphs. *J. Graph Algorithms Appl.*, 15(3):457–482, 2011.

[32] D. Poulalhon and G. Schaeffer. Optimal coding and sampling of triangulations. *Algorithmica*, 46(3-4):505–527, 2006.

[33] J. S. Provan. The complexity of reliability computations in planar and acyclic graphs. *SIAM J. Comput.*, 15(3):694–702, 1986.

[34] M. Sharir and A. Sheffer. Counting triangulations of planar point sets. *Electr. J. Comb.*, 18(1), 2011.

[35] M. Sharir and E. Welzl. Random triangulations of planar point sets. In *Proceedings of the 22nd ACM Symposium on Computational Geometry, Sedona, Arizona, USA, June 5-7, 2006*, pages 273–281, 2006.

[36] M. Sharir, A. Sheffer, and E. Welzl. On degrees in random triangulations of point sets. *J. Comb. Theory, Ser. A*, 118(7):1979–1999, 2011.

[37] A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Inf. Comput.*, 82(1):93–133, 1989.

[38] R. P. Stanley. *Catalan Numbers*. Cambridge University Press, 2015. doi: 10.1017/CBO9781139871495.

[39] W. T. Tutte. The thickness of a graph. *Indagationes Mathematicae*, 25:567–577, 1963.

[40] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

[41] L. G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[42] A. Wigderson. The complexity of the hamiltonian circuit problem for maximal planar graphs. Technical Report 298, Department of EECS, Princeton University, 1982.