Testing Balanced Splitting Cycles in Complete Triangulations
Vincent Despré, Michaël Rao, Stéphan Thomassé

To cite this version:
Vincent Despré, Michaël Rao, Stéphan Thomassé. Testing Balanced Splitting Cycles in Complete Triangulations. Canadian Conference on Computational Geometry (CCCG 2020), Aug 2020, Saskatchewan/Online, Canada. 10.4230/LIPIcs . hal-03059811

HAL Id: hal-03059811
https://hal.archives-ouvertes.fr/hal-03059811
Submitted on 13 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Abstract

Let $T$ be a triangulation of an orientable surface $\Sigma$ of genus $g$. A cycle $C$ of $T$ is splitting if it cuts $\Sigma$ into two noncontractible parts $\Sigma_1$ and $\Sigma_2$, with respective genus $0 < g_1 \leq g_2$. The splitting cycle $C$ is called balanced if $g_1 \geq g_2 - 1$. The complexity of computing a balanced splitting cycle in a given triangulation is open, but seems difficult even for complete triangulations. Our main result in this paper is to show that one can rule out in polynomial time the existence of a balanced splitting cycle when the triangulation is $\varepsilon$-far to have one. Implementing this algorithm, we show that large Ringel and Youngs triangulations (for instance on 22,363 vertices) have no balanced splitting cycle, the only limitation being the size of the input rather than the computation time.

1 Introduction

A splitting cycle on a surface $\Sigma$ of genus at least 2 is a simple cycle (without self-crossing) that allows to cut $\Sigma$ into two parts non-homeomorphic to disks. In a continuous setting, such a curve always exists and is easy to find. In the discrete setting, $\Sigma$ is defined by a combinatorial map $M$ which is a graph embedded on $\Sigma$ such that each face of the graph is an open disk. In this case, a splitting cycle is a simple cycle (a cycle with no repeated vertex) that separates $\Sigma$ into two parts non-homeomorphic to disks. It is no more true that every surface of genus at least 2 has a splitting cycle and it is NP-complete to decide if a given $M$ admits a splitting cycle [3, 2]. However, splitting cycles can be found when $M$ has some additional properties. For instance, simple triangulations (i.e. without loops, cycle of length 2) are believed to have splitting cycles:

$\blacktriangleright$ Conjecture 1 (Barnette (1982) [14, p. 166]). Every simple triangulation of a surface of genus at least 2 has a splitting cycle.
This conjecture is known to be true only in the case of the double torus [9]. It is formulated for triangulations but has been also investigated for combinatorial maps with minimum face-width (the minimum number of faces crossed by a non-contractible curve). It is easy to build a combinatorial map of face-width 2 without splitting cycles. Zha and Zhao [19] conjectured that a face-width of 3 is sufficient to obtain a splitting cycle and proved that 6 is actually enough. Triangulations are a particular case of this second conjecture since any simple triangulation has face-width at least 3.

Recall that a triangulation is called irreducible if none of its edges can be contracted without violating the condition of simplicity. It is easy to see that if \( T \) has a splitting cycle and is obtained by contracting an edge from some \( T' \) then \( T' \) also has a splitting cycle. Thus, it is sufficient to consider irreducible triangulations. Observe also that irreducible triangulations have face-width exactly 3. The number of irreducible triangulations of a given genus being finite [1, 15, 10], it is theoretically possible to check the conjecture for fixed genus. Sulanke gave an algorithm to compute the set of irreducible triangulations of a fixed genus [17] and used it to prove the conjecture for genus 2 with a computer assisted approach [18]. Unfortunately, the number of irreducible triangulations with respect to the genus grows too fast to hope for a brute force proof, even for genus 3.

In this paper we consider only orientable surfaces \( \Sigma \) of genus \( g \). Therefore, a splitting cycle \( C \) cuts \( \Sigma \) into two parts of respective genus \( g_1, g_2 \), where \( g_1 \leq g_2 \). We call \( g_1 \) the type of \( C \), and \( C \) is called balanced if \( g_1 \geq g_2 - 1 \) (if such a cycle exists for \( T \), we also say that \( T \) is balanced).

It was independently conjectured by Zha and Zhao [19] and Mohar and Thomassen [14, p. 167] that a triangulation (or a combinatorial map of face-width at least 3) have all the possible types of splitting cycles. However, Despré and Lazarus [4] disproved this by showing that some triangulations of complete graphs do not have all the possible types of splitting cycles. More precisely they could certify that some triangulation of \( K_{19} \) or \( K_{43} \) are not balanced. However, the algorithm they use could no rule out the existence of balanced large complete triangulations which still could be 'smoother' than small ones and allow all types of splitting cycles. The key-result of this paper is first to show that existence of balanced cycle in a complete triangulation \( T \) of \( K_n \) can be property-tested, and then to provide an efficient implementation of this algorithm to test large Ringel-Youngs triangulations.

Observe that every splitting cycle \( C \) of a complete triangulation \( T \) of \( K_n \) partitions the edges into three classes \((R, L, C)\), where \( C \) are the edges of the cycle, \( R \) the edges to the right of \( C \), and \( L \) the one to the left. Moreover, in the cyclic order \( \sigma_v \) induced by \( T \) around the edges incident to each vertex \( v \), the order of the types of edges is \((R, C, L, C)\). In particular, we never have the cyclic pattern \( R, L, R, L \). This allows a relaxation of the notion of splitting cycle. Precisely, for every \( \varepsilon > 0 \), an \( \varepsilon \)-cycle of \( T \) is a partition of the edges into three classes \((R, L, U)\) such that:

- No vertex \( v \) have the cyclic pattern \( R, L, R, L \) in \( \sigma_v \).
- All but \( \varepsilon n \) of the vertices \( v \) of \( T \) are typical, i.e. every cyclic interval of \( \sigma_v \) of length \( \varepsilon n \) contains an edge \( R \) or an edge \( L \).

We say that an \( \varepsilon \)-cycle \((R', L', U)\) approximates a splitting cycle \((R, L, C)\) if \( R' \subseteq R \) and \( L' \subseteq L \) (here \( U \subseteq C \) and stands for unknown). Our main result is the following:

**Theorem 2.** There is a randomized algorithm running in time \( f(\varepsilon)\text{poly}(|T|) \) which takes as input a complete triangulation \( T \) and returns w.h.p. a set \( X \) of \( \varepsilon \)-cycles such that every splitting cycle of \( T \) is approximated by some element of \( X \). Moreover, the size of \( X \) only depends on \( \varepsilon \).
Note that if $T$ has a balanced cycle $C$, then the previous algorithm will find w.h.p. a balanced $\varepsilon$-cycle (in a sense to be defined later). Let us say that $T$ is $\varepsilon$-far to be balanced if it does not have a balanced $\varepsilon$-cycle. We have the following corollary:

**Theorem 3.** There is a randomized algorithm running in time $f(\varepsilon)\text{poly}(|T|)$ which takes as input a complete triangulation $T$ which is either balanced or $\varepsilon$-far to be balanced and returns w.h.p. either a balanced $\varepsilon$-cycle, or a certificate that no balanced cycle exists.

The previous algorithms are based on sampling a good set of vertices and can indeed be derandomized. However, even in the randomized version, the size of the family $X$ is too large to allow any practical use. Luckily, when restricted to finding a set $X$ approximating every balanced splitting cycle (hence cutting branches leading to unbalanced cycles), it turns out that a mix of random sampling and greedy choices can be implemented in a more efficient way. We could use this implementation in order to rule out the existence of balanced cycles in large Ringel and Youngs triangulations.

The fact that all splitting cycles can be $\varepsilon$-approximated by a bounded set $\Sigma$ is non-intuitive if we think of the continuous setting. Indeed, the number of homotopy classes corresponding to balanced splitting cycles is infinite on the surface of genus $g$. By fixing a natural constant curvature metric on the underlying surface, it is known that the number of homotopy classes corresponding to splitting cycles that can be realized with length at most $L$ is asymptotically $L^{6g-6}$ [13]. In the discrete setting, we cannot reach an infinite number of homotopy classes since we only have a finite number of simple cycles. However, it would have been natural to expect a $K(g)$ (and thus $n$) dependency for the size of $X$.

The problem of constructing triangulations of complete graphs is a very classical one, raised by Heawood in 1890 [8]. The original aim was to find an optimal proper coloring of a graph embedded on a surface of genus $g > 0$. Apart from the case of the sphere (or the plane) and the Klein bottle, the Euler formula already gives the exact upper bound of $\gamma(g) = \left\lfloor \frac{7+\sqrt{49+48g}}{2} \right\rfloor$ colors. Hence, to prove the tightness of the bound, it was necessary to produce a graph of genus $g$ with chromatic number $\gamma(g)$. This has been achieved by Ringel and Youngs [16, 7] using complete graphs. The embeddings they provided are minimal in the sense that each complete graph cannot be embedded on a smaller genus surface and some of them are triangulations. Actually, there are many different triangulations of a given complete graph [12, 11, 6, 5]. For the experiments in this paper we will focus on the triangulations given by Ringel and Youngs for $n \equiv 7[12]$.

The major difficulty here is that the size of the sample which gives the certificate is too large to allow computation based on a one-step guess. We instead adopt a randomized greedy strategy in order to iteratively construct the sample. The algorithm is described in details in Section 5. This algorithm is extremely efficient and allow to address huge triangulations. Actually, it may be used as soon as the size of the triangulation can be stored on the computer. It has been implemented independently by Vincent Despré and Michaël Rao and they were able to reach very huge complete triangulations.

**Theorem 4.** The complete triangulation with 22,363 vertices (and 250,040.703 edges) given by Ringel and Youngs has no balanced splitting cycle.

The implementations details along with the different results are developed in Section 6. Our algorithm is a new tool to deal with splitting cycles and may be useful in a larger spectrum. Indeed, when it fails to prove that the input triangulation has no balanced splitting cycles, it gives hints to find possible ones since it outputs balanced $\varepsilon$-cycles which can be the seed of some new investigation. This is probably the most appealing open question.
Testing Balanced Splitting Cycles in Complete Triangulations

left by the paper: Given a balanced ε-cycle, how to decide if it can be extended or not into a balanced (or near balanced) cycle. If one could design an efficient algorithm in order to find balanced splitting cycles, it would lead to efficient divide and conquer algorithms on complete triangulations.

We first describe the background and notations in Section 2 and give some technical results about the structure of splitting cycles in Section 4.

2 Notations and Background

Combinatorial surfaces

As usual in computational topology, we model a surface by a cellular embedding of a simple graph G (without loops or multiple edges) in a compact topological surface Σ. Such a cellular embedding can be encoded by a combinatorial surface composed of the graph G itself together with a rotation system [14] that records for every vertex of the graph the clockwise order of the incident edges. The facial walks are obtained from the rotation system by the face traversal procedure as described in [14, p.93]. We denote by n, e and f the numbers of vertices, edges and faces of the combinatorial surface. The genus g of Σ is linked to the embedding via a very strong topological property that we call Euler characteristic:

χ(Σ) = n − e + f = 2 − 2g. A triangulation is a particular kind of combinatorial map whose all faces are triangles. The combinatorial maps that we consider in this paper consists of triangulations of complete graphs where a complete graph is a graph containing all the possible pairs of vertices as edges. Such a triangulation does not trivially exists. It requires that n ≡ 0, 3, 4 or 7[12] and even in this case the constructions are not straightforward. We consider a triangulation Tn for theoretical construction but the experiments are only done for the triangulations given by Ringel and Youngs [16] for n = 12s + 7. To summary, we have that the rotation scheme around the vertex vi is a cyclic permutations σi of \{1,...,n\} \ i, such that: for every triangle ijk, if k is the successor of j in σi, then i is the successor of k in σj.

Data-structure

To be able to correctly analyze the complexity of our algorithm, it is necessary to describe a bit the data-structure we use: the half-edge data-structure. It consists in coding Tn by a set of half-edges each having an handle to the opposite half-edge (represented by an involution α0) and to the next half-edge in the local σi (we can think of it as a global permutation σ whose cycles are the σi). At this point, we can notice that the size of the map is actually 2e·<size of an half-edge>= O(e). An edge is an orbit of the action of α0 on the set of half-edges and can be stored as one element in the orbit. Similarly, the orbits of σ are the vertices, it is again sufficient to store one half-edge for each vertex. We need to store on each vertex a 'reverse' dictionary Revi that associate to every vertex vj for j \ i its position around vi (each vertex is associated to a unique half-edge around vi). The Revi dictionaries are not a general feature in the half-edge data-structure but is required by our algorithm. Finally, the faces can be construct by alternatively applying α0 and σ and storing a half-edge for each corresponding orbit. Here, computing the faces is mainly useful to check that Tn is a correct triangulation. The construction of the map is considered as a precomputation and is clearly done using O(e) operations.
Combinatorial curves

Consider a combinatorial surface with its graph $G$. A cycle $C$ is a closed walk in $G$ without repeated vertex. $C$ may have different topological types. To understand this we need to define a (free) homotopy. Two closed continuous curves on $\Sigma \alpha, \beta : \mathbb{R}/\mathbb{Z} \rightarrow \Sigma$ are homotopic, if there exists a continuous map $h : [0, 1] \times \mathbb{R}/\mathbb{Z}$ such that $h(0, t) = \alpha(t)$ and $h(1, t) = \beta(t)$ for all $t \in \mathbb{R}/\mathbb{Z}$. Intuitively it means that two curves are homotopic if one can be continuously deformed into the other. We say that $C$ is contractible if it is homotopic to a point and separating if its removal leaves two connected components on $\Sigma$. $C$ may have three different homotopy types which are: contractible and separating, non-contractible and non-separating and non-contractible and separating (see Figure 1). If $C$ is of this last type then we called it a splitting cycle. We can refine the notion of homotopy types for splitting cycle. Indeed, the genera $g_1$ and $g_2 \geq g_1$ of the two connected components defined by a splitting cycle are additive in the sense that $g_1 + g_2 = g$ (this is a direct consequence of the Euler characteristic which becomes $\chi(\Sigma) = 2 - 2g - b$ if $\Sigma$ has $b$ boundaries). We define the type of a splitting cycle as the genus $g_1$. For instance, a splitting cycle of type 1 cuts $\Sigma$ into a torus with one boundary and a surface of genus $g-1$ with one boundary. We say that $C$ is balanced if it has type $\lfloor \frac{g}{2} \rfloor$.

\[ \text{Figure 1} \ C_1 \text{ is contractible, } C_2 \text{ is a splitting cycle and } C_3 \text{ is non-separating.} \]

Approximation of cycles

As said before, a splitting cycle $C$ of $T_n$ induces an edge coloring of the edges of $K_n$ into colors $(L, R, C)$ such that $C$ is the cycle and no $R$ and $L$ edges are cyclically adjacent in $\sigma_v$ for all $v$. We will now see the splitting cycle $C$ as the partition $(L, R, C)$. In particular, when $C$ is balanced, this translates a sparse object (the splitting cycle $C$) into a dense object (the edge coloring $(L, R, C)$) since both $R$ and $L$ have quadratic size. This allows approximation of $L$ and $R$ by sampling. We now say that $(L', R', U)$ approximates $(L, R, C)$ if $R' \subseteq R$ and $L' \subseteq L$. The partition $(L', R', U)$ is an $\varepsilon$-cycle if:

- For every vertex $v_i$, the cyclic order $\sigma_i$ does not contain four cyclically ordered edges $R', L', R', L'$.
- All but $\varepsilon n$ of the vertices $v$ of $T$ are typical, i.e. every cyclic interval of $\sigma_i$ of length at least $\varepsilon n$ contains an edge of $R'$ or an edge of $L'$.

3 Efficiently approximating cycles

Our goal is to prove Theorem 2, which shows that one can efficiently find a set $X$ of $\varepsilon$-cycles approximating all splitting cycles.
Theorem 5. For every orientable triangulation \( T_n \) of \( K_n \) and every \( \varepsilon > 0 \), there is a set \( X \) of size \( f(\varepsilon) \) consisting of \( \varepsilon \)-cycles such that every splitting cycle of \( T_n \) is approximated by some element of \( X \).

Proof. Pick some large constant \( c > 4/\varepsilon^2 \). We implicitly assume here that \( n \) is much larger than \( \varepsilon \) and \( c \), otherwise \( X \) simply exists by enumeration. Pick uniformly at random a sample \( S \) of vertices of \( T_n \) of size \( c \). For each \( v_i \in S \), divide the cyclic order \( \sigma_i \) into \( c \) cyclic intervals \( I_1, \ldots, I_c \) of approximately the same length (i.e. size \( \lfloor (n-1)/c \rfloor \) or \( \lceil (n-1)/c \rceil \)). We now construct our \( \varepsilon \)-cycles \((R, L, U)\). We first decide for each \( v_i \in S \) an \( R, L, U \) (right, left, unknown) coloring of the intervals \( I_j \) in such a way that two (possibly identical) intervals are \( U \) and these two \( U \) intervals separates the \( R \) intervals and the \( L \) intervals. Note that when the \( U \) intervals are identical or adjacent, the remaining intervals are all colored \( R \) or all colored \( L \). The total number of such choices for a given \( v_i \in S \) is \( c^2 + c \). And we then have \((c^2 + c)^c \) possible ways of coloring the edges adjacent to \( S \) according to this local rule.

Among these coloring, some of them are inconsistent in the sense that they give both colors \( R \) and \( L \) at the two endpoints of some edge between two elements of \( S \). We reject these colorings. It can also happen that an edge receives both colors \( U \) and \( R \) (or \( U \) and \( L \)) in which case the edge keeps the color different from \( U \). We then color \( U \) all edges which were not incident to vertices of \( S \). We reject all colorings which contain the forbidden pattern \((R, L, R, L)\) in some \( \sigma_i \). The set of surviving \((R, L, U)\) colorings is denoted by \( X_S \), and this is our candidate for \( X \). Note that the size of \( X_S \) only depends on \( c \) and hence on \( \varepsilon \), and that the total number of \( U \) edges incident to points of \( S \) is at most \( c.2n/c \).

The key-observation is that every splitting cycle \( C \) of \( T_n \) is approximated by some element of \( X_S \). Indeed, for each vertex \( v_i \in S \) one can define the two \( U \) intervals of \( \sigma_i \) as these containing an edge of \( C \), and the \( R \) and \( L \) intervals are the one which are entirely \( R \) or \( L \) according to cycle \( C \). So to reach our conclusion, we just have to show that every element of \( X_S \) is an \( \varepsilon \)-cycle.

We claim that this happens if we are lucky enough with our sampling \( S \). Let us say that a vertex \( v_i \) is good if \( S \) is well distributed in \( \sigma_i \). More precisely if for every cyclic interval of \( \sigma_i \) of size at least \( \varepsilon n \), the number of elements of \( S \) is at least \( \varepsilon c/2 \). Observe that the probability that a vertex is good tends to 1, when \( \varepsilon \) is fixed and \( c \) goes to infinity. By Markov, we can fix \( c \) large enough such that with high probability, our sampling \( S \) will be such that all vertices save an arbitrarily small proportion are good. We now claim that in this case, all \((R, L, U)\) partitions of \( X_S \) are \( \varepsilon \)-cycles.

Assume for contradiction that this is not the case. Then there are more than \( \varepsilon n \) non typical vertices \( v_i \) for which \( \sigma_i \) contains an interval \( I_{j_i} \), of size at least \( \varepsilon n \) with no \( R \cup L \) edge. Since we can neglect these vertices \( v_i \) which are either in \( S \) or non good vertices, each of these intervals \( I_{j_i} \) contains \( \varepsilon c/2 \) vertices of \( S \), and none of them have created an \( R \cup L \) edge with \( v_i \). So the total number of \( U \) edges incident to vertices of \( S \) is at least \( \varepsilon n.\varepsilon c/2 \), which is contradicting the fact that there are at most \( c.2n/c \) of them since \( c > 4/\varepsilon^2 \).

This concludes the proof of Theorem 2, the algorithm simply returning \( X_S \) for some large enough sample \( S \). The main drawback of this approach is the size of the sampling, which makes it very difficult to implement for some practical use. Since our goal is to look for balanced splitting cycles, we will only focus on \( \varepsilon \)-cycles which can be approximations of balanced cycles. Let us denote by \( tr(n) \) the minimum size of \( R \) (or equivalently of \( L \)) in a balanced cycle \((R, L, C)\) of an orientable triangulation of \( K_n \). Note that \( tr(n) = n^2/4 - O(n) \), but a more precise value will be given later when we will discuss the implementation. Thus if some \( \varepsilon \)-cycle \((R', L', U)\) approximates \((R, L, C)\), it must have potentially at least \( tr(n) \).
many $R'$ or $L'$ edges. Let us properly define this. The right-potential $r(v_i)$ of some vertex $v_i$ is defined as:

- When $v_i$ is incident to some edges of $R'$ and $L'$, $r(v_i)$ is the size of the longest cyclic interval of $\sigma_i$ with a point in $R'$ and no point in $L'$, minus 2.
- When $v_i$ is only incident to edges of $R'$, we have $r(v_i) = n - 1$.
- When $v_i$ is only incident to edges of $L'$, $r(v_i)$ is the size of the longest cyclic interval of $\sigma_i$ with no point in $L'$, minus 2.

The same definition applies for left potential $l(v_i)$. The right-potential $r(R', L', U)$ is the sum of the right potential of all the vertices (same for left-potential $l(R', L', U)$). Note that $r(R', L', U) \geq 2|R|$ and $l(R', L', U) \geq 2|L|$ when $(R', L', U)$ approximates $(R, L, C)$ (the factor 2 in the inequality stands for the fact that we are doubly counting edges in the potential). Let us then say that an $\varepsilon$-cycle $(R', L', U)$ is unbalanced if $r(R', L', U) < 2\varepsilon n$ or $l(R', L', U) < 2\varepsilon n$ (otherwise it is balanced). A triangulation $T_n$ is $\varepsilon$-far to be balanced if it has no balanced $\varepsilon$-cycle.

**Proof of Theorem 3.** Now let us prove that we can efficiently separate triangulations which are either balanced or $\varepsilon$-far to be balanced. For this, we compute a set $X_S$ of $\varepsilon$-cycles which approximates all splitting cycles of $T_n$. Note that if $T_n$ admits a balanced cycle $(R, L, C)$, then it is approximated by some $\varepsilon$-cycle $(R', L', U)$ in $X_S$ which hence must be balanced and thus a certificate of separation. Now if $T_n$ does not admit a balanced cycle $(R, L, C)$, we compute a set $X_S$ coming w.h.p. from a lucky sample $S$. The key point is that we can indeed check if $S$ is a good sample or not, just by checking if it is well-distributed in nearly all $\sigma_i$. Hence the set $X_S$ probably approximate all splitting cycles of $T_n$, and if we satisfy the separation hypothesis of Theorem 3, none of the $\varepsilon$-cycles are balanced. Therefore $X_S$ is a certificate of the fact that $T_n$ has no balanced splitting cycle. ▷

The nice feature of this property-testing algorithm is that if we try to check if a given $T_n$ has a balanced cycle, we may be lucky and get a NO-certificate. This is basically what happens so far for all Ringel and Youngs triangulations on which the algorithm terminates. However, in the present form, the size of $X_S$ is way too large to be implemented, and we will use a mix of random sampling and greedy choices for $S$. Also the fact that we divide $\sigma_i$ into $c$ intervals is convenient for the proof but not for the algorithm, which will only cut into 3 parts.

Another exciting direction of research is when we get a set $X_S$ of $\varepsilon$-cycles, some of which being balanced. There is possibly a way to investigate if a given balanced $\varepsilon$-cycle can be completed into a balanced (or near balanced) cycle. For instance, if some $\sigma_i$ contains the pattern $(R, U, R, L)$, then the $U$ edge can be turned into an $R$ edge (possibly creating forbidden patterns leading to reduction of $X_S$). These closure operations (together with a $(L, U, L, R)$ rule) can greatly densify our candidate $\varepsilon$-cycle making it easier to complete or not into a splitting cycle.

### 4 Properties of Splitting Cycles of Complete Triangulations

We begin by fixing some specific notations. We need to split the neighborhood of the vertices into parts. Mainly, if $v_i$ is a vertex we denote by $\{ev_0, ev_1, ev_2\}_i$ a partition of the vertices (or equivalently of the half-edges) around $v_i$ such that the edges of each $ev_k$ is consecutive with respect to $\sigma_i$. We call a local configuration a couple $(i, c)$, where $i$ corresponds to the part $ev_i$ and $c$ is a color and a configuration a list of local configurations.
Lemma 6. Let v be a vertex of $T_n$, $(ev_0, ev_1, ev_2)$ be any partition of the edges in the neighborhood of v and $(L, R, C)$ be a splitting cycle of $T_n$. At least one of the $ev_i$ is entirely colored L or R.

Proof. C may reach at most two of $ev_0$, $ev_1$ and $ev_2$. It implies that one of the $ev_i$ has to be colored entirely L or R for any splitting cycle.

We thus obtain 6 different configurations for v. The following lemma is a direct consequence of the previous one.

Lemma 7. Let $(v_0, \cdots, v_k)$ be a list of vertices of $T_n$ and $(ev_0, ev_1, ev_2)$ be a fixed partition of the edges around $v_i$, for all $0 \leq j < k$. Then, there is a configuration $((i_0, c_0)v_0, \cdots, (i_{k-1}, c_{k-1})v_{k-1})$ realized by each splitting cycle $(L, R, C)$.

Let us now consider the particular properties of balanced splitting cycles of complete triangulations.

Lemma 8. Let $C = (L, R, C)$ be a balanced splitting cycle of $T_n$. Then,

$$|C| \geq \left[ \frac{5 + \sqrt{2n^2 - 14n + 25}}{2} \right]$$

$$tr(n) = \min(|L|, |R|) \geq \left[ \frac{n^2 - 7n + 8 + 4\sqrt{2n^2 - 14n + 25}}{4} \right]$$

Proof. Since we consider complete graphs, it is not possible that there exists two vertices colored entirely R for one and L for the other one. Hence, after cutting along C, there is a map with one boundary and no interior vertex of genus at least $\left[ \frac{G}{2} \right]$. Let $k = |C|$ and $T'$ be the map without interior vertices obtained after cutting along C. $T'$ has genus at least $\left[ \frac{G}{2} \right]$ and so $\chi(T') \leq 2 - 2 \left[ \frac{G}{2} \right] - 1 \leq 2 - (g - 1) - 1 = 2 - g$. $M'$ has k vertices, $e \leq \frac{k(k-1)}{2}$ edges and $f$ faces. The double counting of the number of edges gives $3f = 2e - k$ because all the edges are on exactly 2 faces except the k on the boundary. So $\chi(T') = k - e + \frac{2f}{2} = \frac{2k - \frac{k(k-1)}{2}}{2} \geq \frac{4k - k(k-1)}{6} = \frac{5k - k^2}{6}$. By putting together the two inequalities we obtain: $2 - g \geq \frac{5k - k^2}{6}$ leading to $k^2 - 5k + 6 - 6g \geq 0$. $\Delta = 25 - 4(6 - 6g) = 1 + 24g$ and so $k = |C| \geq \frac{5 + \sqrt{1 + 24g}}{2} = \frac{5 + \sqrt{1 + 24(n-3)(n-4)}}{2} = \frac{5 + \sqrt{2n^2 - 14n + 25}}{2}$.

Let us look back at the Euler formula for $T'$. We have, $\chi(T') = \frac{2k - e}{2} \leq 2 - g$. It implies that $e \geq 2k + 3g - 6 \geq 5 + \sqrt{2n^2 - 14n + 25} + \frac{3(n-3)(n-4)}{12} - 6 = \frac{(n-3)(n-4) + 4\sqrt{2n^2 - 14n + 25} - 4}{4}$.

It is interesting to notice that $\frac{e}{\min(|L|, |R|)} = \frac{1}{2} - O\left(\frac{1}{n}\right)$ for balanced splitting cycles in complete triangulations and thus $tr(n) = \frac{n^2}{4} - O(n)$.

5 Algorithm

Sketch

We first describe the sketch of the algorithm. We suppose that a balanced splitting $(L, R, C)$ exists and we want to obtain a contradiction. We choose at random a set of k vertices $(v_0, \cdots, v_{k-1})$ of $T_n$ and $(ev_0, ev_1, ev_2)$ a balanced partition of the edges around $v_i$, for all $0 \leq j < k$. By Lemma 7 we have a configuration $((i_0, c_0)v_0, \cdots, (i_{k-1}, c_{k-1})v_{k-1})$ realized by
V. Despré, M. Rao and S. Thomassé

C. Up to a natural symmetry we can assume that \( c_0 = L \). Thus, we have \( 3 \cdot 6^{k-1} \) possible configurations. We need to show that every configuration is not admissible. We look at the other vertices of the graph, we consider the colors induced by a given configuration and we have two tools to show that the configuration is not correct. First, if we can find an alternated sequence of edges labeled \((L, R, L, R)\) around a vertex, then this vertex violates the conditions of \((L, R, C)\). We can also look the biggest number of edges colored \( R \) that each vertex can admits and use \( tr(n) \) given by Lemma 8 to reject the configuration.

We want to explore the tree of all the possible configurations. We design this tree such that the layer \( i \) corresponds to the choice of the local configuration for the vertex \( v_i \). A first approach is to take \( k \) big enough to reach a contradiction for all leaves of the research tree. This is not reasonable because of the growth of the size of the tree so we decide to check all the nodes in the tree where an internal node corresponds to a partial configuration. If this partial configuration already gives a contre-example then all the subtrees from the corresponding node can be discarded. In addition, we don’t need to use the same \( v_k \) on all the nodes of a given layer. It means that we construct a tree of configurations starting from the root which is the empty configuration and we avoid getting deeper in the tree as soon as the can prove that the corresponding configurations is not correct.

**Algorithm**

**INPUT:** A complete triangulation.

- Let \( C \) be an empty vector of configurations. We initialize \( RandV \) with a random vertex \( v_i \) and a random partition of the neighborhood of \( v_i \) into three consecutive parts \((ev_0, ev_1, ev_2)_i\). We put the configuration \((v_i, (ev_0, ev_1, ev_2)_i, 0, L)\) in \( C \).
- We add a list \( L_j \) on each vector \( v_j \) that stores the position of the vertices already colored.
- At this stage, it means that for all \( v_j \in ev_0 \) we call \( Rev_j(i) \) to know the position of \( v_i \) around \( v_j \) and we put \((Rev_j(i), L)\) in \( L_j \). Notice that the \( L_i \)s must be sorted during the algorithm.
- While \( C \) is non-empty we do:
  1. We test if \( C \) is valid. This implies two tests:
     - We look at all the \( L_i \)s to see if there is no cyclic subsequence of the form \((L, R, L, R)\).
     - We sum the biggest interval that can be colored \( L \) (resp. \( R \)) in all the \( L_i \)s and we compare the result to the one of Lemma 8.
  2. If one of the test fails we update \( C \) in the following way:
     - If the last element of \( C \) is of the form \((\cdots, 2, R)\) then we discard it and we update \( C \) again.
     - Else we consider the next configuration using the order: \((0, L), (1, L), (2, L), (0, R), (1, R)\) and \((2, R)\).
     We update the \( L_i \)s to make it coherent with the new configuration and the go back to step 1.
  3. We compute a new random vertex \( v_i \) not already used by \( C \) with a partition of its neighborhood and we add \((v_i, (ev_0, ev_1, ev_2)_i, 0, L)\) at the end of \( C \). We then update the \( L_i \)s and go back to 1.

**Analysis of the algorithm**

**Theorem 9.** If the algorithm terminates then the input triangulation does not have a balanced splitting cycle.
The algorithm described above requires $O(t \cdot d \cdot n) = O(t \cdot d \cdot \sqrt{T})$ operations where $t$ is the size of the research tree $T$ and $d$ its depth.

**Proof.** Each node of $T$ corresponds to one turn in the While. Step 1 requires to read all the lists $L_i$. There are $n$ such lists and their size is bounded by the size of $C$ which is less than the depth of $T$. It implies that this step requires $O(d \cdot n)$ operations. Step 2 and 3 may require an insertion or a deletion in one third of the $L_i$, which is clearly done in $O(d \cdot n)$ operations. Since we consider $t$ configurations, we obtain a total of $O(t \cdot d \cdot n)$ operations. ▶

**Optimizations**

When we reach some depth in $T$ it becomes interesting to choose smartly the next local configuration. Indeed, if the new vertex $v_i$ already has two half-edges colored $L$ pointing to vertices $v_0$ and $v_1$, then it should be interesting to consider a local configuration $(ev_0, ev_1)$, such that $ev_0$ and $ev_1$ are delimited by $v_0$ and $v_1$. Indeed, one of this two sets have to be entirely colored $L$ in a $(L, R, C)$ splitting cycle. We obtain only two local configurations to check $(0, L)$ and $(1, L)$ instead of 6. To be sure that the $ev_0$ and $ev_1$ both contain enough edges, we only use this setting when we find two half-edges of the same color separated by at least a fixed distance $p.e$ around $v_i$. After some testing, we decided to set $p$ to 0.35, this parameter may be changed but should stay in an interval $[0.3, 0.45]$ to be useful. In addition, some minor optimizations can be made, we can check the $(L, R, L, R)$ conditions while we update the $L_i$s for instance.

The algorithm is highly parallelizable since different subtrees can use uncorrelated vertices. The parallelization works as follows: we first set a value $d_0$ as the initial depth in the tree of research and we choose $d_0$ fixed random vertices, then we set a list of tasks corresponding to the $3 \cdot 6^{d-1}$ leaves of the initial tree. Now a master thread send a configuration corresponding to a leaf to every other threads as soon as they achieved their previous task.

To prove that a configuration is impossible, a thread may need to construct its own subtree, thus we decide to give to each thread a different copy of the data-structure. To reach bigger triangulations, it may be useful to use a unique copy on each node but this will require to put it read-only and so extract the $L_i$ form the data-structure which may represent a loss of performance.

**Implementation details and experimental results**

The implementation has been realized in C++ using OPENMPI for parallelization and can be downloaded at http://vdespre.free.fr/Splitting.tar.gz. The test had been launched on the cluster Grid’5000. Let $m$ be the number of threads for given experiment. The choice of $d_0$ can be optimized for each case so we precise what we used in each case.

---

1 Experiments presented in this paper were carried out using the Grid’5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).
We first give results to show the efficiency of the algorithm. Notice that the limit is set by the RAM on each node and so the number of threads is set to not break the memory limit. The time column shows the average on 10 trys.

<table>
<thead>
<tr>
<th>s</th>
<th>n</th>
<th>e</th>
<th>time (s.)</th>
<th>CPU time</th>
<th>m</th>
<th>nodes</th>
<th>$d_0$</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>833</td>
<td>1003</td>
<td>50025003</td>
<td>425</td>
<td>21h15m</td>
<td>180</td>
<td>45</td>
<td>6</td>
<td>2 000 000</td>
</tr>
<tr>
<td>1863</td>
<td>22363</td>
<td>250040703</td>
<td>2990</td>
<td>37h22m</td>
<td>45</td>
<td>45</td>
<td>5</td>
<td>1 700 000</td>
</tr>
</tbody>
</table>

It is interesting to notice that the time of the tests highly depends on the exact value of $n$. It means that the size of the research tree is not smooth with respect to $n$. It is pretty surprising and we have no hint of the reason by now. The following experiments have been done using 720 threads on 45 nodes.

<table>
<thead>
<tr>
<th>s</th>
<th>n</th>
<th>time (s.)</th>
<th>$\sigma$ (s.)</th>
<th>$d_0$</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1207</td>
<td>18</td>
<td>1</td>
<td>7</td>
<td>1 800 000</td>
</tr>
<tr>
<td>101</td>
<td>1219</td>
<td>62</td>
<td>15</td>
<td>7</td>
<td>2 100 000</td>
</tr>
<tr>
<td>102</td>
<td>1231</td>
<td>945</td>
<td>224</td>
<td>9</td>
<td>41 000 000</td>
</tr>
<tr>
<td>103</td>
<td>1243</td>
<td>970</td>
<td>178</td>
<td>9</td>
<td>42 000 000</td>
</tr>
<tr>
<td>104</td>
<td>1255</td>
<td>17</td>
<td>1</td>
<td>7</td>
<td>1 800 000</td>
</tr>
<tr>
<td>105</td>
<td>1267</td>
<td>fails in 7200</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>1279</td>
<td>35</td>
<td>8</td>
<td>7</td>
<td>1 900 000</td>
</tr>
<tr>
<td>107</td>
<td>1291</td>
<td>42</td>
<td>4</td>
<td>7</td>
<td>1 900 000</td>
</tr>
<tr>
<td>108</td>
<td>1303</td>
<td>220</td>
<td>45</td>
<td>7</td>
<td>8 200 000</td>
</tr>
<tr>
<td>109</td>
<td>1315</td>
<td>17</td>
<td>1</td>
<td>7</td>
<td>1 800 000</td>
</tr>
<tr>
<td>110</td>
<td>1327</td>
<td>18</td>
<td>1</td>
<td>7</td>
<td>1 800 000</td>
</tr>
</tbody>
</table>

### Conclusion

The structure of the splitting cycles in triangulations of complete graphs remains quite mysterious. Even for the case of Ringel and Youngs embeddings restricted to $n = 12s + 7$, we do not understand what exactly happens. Our new experimental results give some informations on the absence of balanced splittings. In this specific case, we can imagine to make tests on bigger triangulations by storing the embedding using $O(n)$ memory. This can be done using the extreme symmetry of the embeddings but is not likely to be generalized.

We can also want to explore other triangulations of complete graphs. A very simple question remains open on this subject:

**Question 11.** Is there an unbounded sequence of triangulations of complete graphs admitting balanced splitting cycles?

The question is of intrinsic interest and it is difficult to have an intuition about it. The constructions of triangulations of complete graphs are pretty intricate and it is not clear if one can be modified to ensure the existence of a balanced splitting. In addition, we always look for an easy proof that some triangulation does not have a splitting cycle. We think that Theorem 5 is the kind of idea that can lead to such a proof. However, it is not clear how much the properties of a specific embedding must be used. In case that there exists huge triangulations of complete graphs with balanced splittings, embeddings become critical. If not, we can imagine to prove the non-existence of balanced splitting in complete triangulations without considering a specific embedding which is very convenient, in particular for probabilistic arguments.
Testing Balanced Splitting Cycles in Complete Triangulations

References


