Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Marek Krčál

Výpočetní složitost testování rovinnosti grafu

Katedra aplikované matematiky

Vedoucí bakalářské práce: Mgr. Martin Bálek, Institut teoretické informatiky
Studijní program: matematické struktury

2006

Acknowledgement: I thank prof. Eric Allender for proposing the me to try to solve the problem by devising a reduction to the degree three graphs and for advising my thesis.

I also thank Martin Bálek for giving me many useful notes on my thesis.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne Marek Krčál

Contents

Název práce: Výpočetní složitost testování rovinnosti grafu
Autor: Marek Krčál
Katedra (ústav): Katedra aplikované matematiky
Vedoucí bakalářské práce: Mgr. Martin Bálek
e-mail vedoucího: balek@kam.mff.cuni.cz

Abstrakt: V tomto článku ukážeme, že testování planarity je v SL (symetrický nedeterministický LOGSPACE). Hlavní část našeho důkazu je redukce na problém testování rovinnosti grafu s maximálním stupněm tři. Povšiměte si, že obvyklé nahrazování vrcholů větších stupňů "malými kružnicemi" může rovinnost pokazit, musíme si počínat šikovněji. Testování rovinnosti grafu s maximálním stupněm tři už bylo vyřešeno ve článku "Symmetric complementation" Johna Reifa.

Už dříve Meena Mahajan a Eric Allender ("Complexity of planarity testing") ukázali, že testování rovinnosti je v SL. Jejich důkaz se však sestává z SL implementace velmi složitého paralelního algoritmu od Johna Reifa a Vijayi Ramachandran ("Planarity testing in parallel"). Ten je však nejspíše zbytečně komplikovaný pro účely prostorové složitosti.

Tento výsledek spolu s nedávným průlomem Omera Reingolda dokazujícího, že SL = L ("Undirected ST-connectivity in log-space") zcela řeší otázku složitosti testování planarity, protože to je těžké pro L (toto je též dokázáno v "Complexity of planarity testing"). Zkonstruujeme algoritmus používající logaritmický prostoru, který převede vstupní graf G na G' s maximálním stupněm 3 tak, že že G je rovinný tehdy a jen tehdy, když G' je rovinný.

Klíčová slova: planarita grafu, LOGSPACE, složitost

Title: Computational Complexity of Graph Planarity Testing
Author: Marek Krčál
Department: Department of applied mathematics
Supervisor: Mgr. Martin Bálek
Supervisor's e-mail address: balek@kam.mff.cuni.cz

Abstract: In this paper we will show that the problem of planarity testing is in SL (symmetric nondeterministic LOGSPACE). The main part of our proof is a reduction of the problem to planarity of graphs with maximal degree three. Note that usual replacing vertices of degree bigger than three by "little circles" can spoil planarity, we need to be smarter. Planarity of graphs with maximal degree three was already solved in paper "Symmetric complementation" by John Reif.

Previously Meena Mahajan and Eric Allender have already proved this in ("Complexity of planarity testing"), but their proof is the pure SL implementation of a parallel algorithm by John Reif and Vijaya Ramachandran ("Planarity testing in parallel"). But it is possibly unnecessarily complex and sophisticated for the purposes of the space complexity.

This result together with recent breakthrough by Omer Reingold that SL = L ("Undirected ST-connectivity in log-space") completely solves the question of complexity of planarity problem, because planarity is hard for L (it is again shown in "Complexity of planarity testing"). We construct logarithmic-space computable function that converts input graph G into G' with maximal degree three such that G is planar if and only if G' is. This together with

Keywords: graph planarity, LOGSPACE, complexity

Chapter 1

Introduction

The problem of determining if a graph is planar has been studied from several perspectives of algorithmic research. We focus on space complexity of the problem: we will show that it lies in SL class. This together with very recent result [Rei05] stating that $\mathsf{SL} = \mathsf{L}$ completely solves complexity of planarity testing, because in [AM04] is shown, that it is hard for L under projection reducibility. (L denotes problems decidable by algorithms that involve only logarithmic amount of memory.)

This is the same result as one given by [AM04], but we hopefully provide more intuitive proof, than was pure SL implementation of a highly efficient parallel algorithm by Ramachandran and Reif [RR94], which is probably unnecessarily complicated for purposes of SL. We give only $\mathsf{FL}^{\mathsf{SL}}$ reduction to already SL-solved problem ([Rei84]) of graphs of maximal degree three. This idea come out of the advisor of this work Eric Allender and is put down in introductory summary of his paper together with Meena Mahajan [AM04]:

> In a recent survey of problems in the complexity class SL [AG00], the planarity testing problem for graphs of bounded degree is listed as belonging to SL, but this is based on the claim in [Rei84] that checking planarity for bounded degree graphs is in the "Symmetric Complementation Hierarchy", and on the fact that SL is closed under complement [NTS95] (and thus this hierarchy collapses to SL). However, the algorithm presented in [Rei84] actually works only for graphs of degree 3, and no straightforward generalization to graphs of larger degree is known. (This is implicitly acknowledged in [RR94, pp. 518,519].)

So we give the generalization to general biconnected graph.

Why is it sufficient to work only with biconnected graph? It is well known fact, that an arbitrary graph G is planar if and only if all its components of connectivity are planar. And because all components of biconnectivity can be found in $\mathsf{FL}^{\mathsf{SL}}$ ([AM04]), algorithm for biconnected graph can be extended for general graph.

Summerized, our goal is to show a $\mathsf{FL}^{\mathsf{SL}}$ function (and its algorithm), which converts given biconnected graph to a reduced graph with maximum degree 3 or rejects. If it rejects, the input graph is not planar, otherwise the reduced graph is planar, if and only if the original graph was planar.

Our work is organized as follows:

Chapter 2 contains all necessary background for our work. Section 1 gives basic definitions from the complexity theory. Section 2 introduces notation and concepts from the graph theory important for our task.

Chapter 3 section 1 contains general explanation how the whole algorithm works. Then follow sections with detail description of all steps of the algorithm including proofs of their correctness.

Chapter 2

Preliminaries

2.1 Complexity background

Throughout we only use some well known facts from the theoretic complexity. But intent of this work is not to study the theoretic complexity classes. We only use them for defining the problem and to restrict algorithms which we can use. Therefore we give olny basic definitions needed and refer the reader interested in complexity theory to some of many books as [Sip96].

Definition 2.1. *By* **L**=**LOGSPACE** *we denote the class of decision problems solvable by a Turing machine restricted to use an amount of memory logarithmic in the size of the input, n. (The input itself is not counted as part of the memory.)*

By **NL** *we denote the class of decision problems solvable by a nondeterministic Turing machine restricted to use an amount of memory logarithmic in the size of the input. An important NL complete problem is the reachability problem for directed graphs (is there a path from vertex s to vertex t?).*

By **SL** *we denote the class of problems that are logarithmic space reducibile to the reachability problem for undirected graphs.*

By **FL** *we denote class of functions computable on a Turing machine with the same memory restrictions as the class L has.*

By $\mathbf{FL}^{\mathbf{SL}}$ *we denote class of functions computable on a Turing machine with the same memory restrictions as the class L has and with oraculum for deciding the reachability problem for undirected graphs.*

Also note that because of [Rei05] L = SL and also FL = $\mathsf{FL}^{\mathsf{SL}}$ but we will still distinguish between them to denote when an algorithm contains some "non-trivially L implementable part" equivalent to the reachability problem for undirected graphs.

The NL class will not be used in this work. We give its definition to show the most important complexity superclass of L and SL and show its similarity to SL.

2.2 Definitions, notation

Definition 2.2. ***Open ear decomposition*** *of a biconnected graph G starting from adjacent vertices s and t is sequence of paths (called* ***ears****) $(P^0 = \langle s - t\rangle, P^1, \ldots, P^k)$. Every ear P^i, $i > 0$ has the first and the last vertex called* ***endpoints*** *(the remaining are called* **internal vertices***) contained in some ear with lower index number and every other vertex of P^i is not contained in any ear with lower index number. Ears which have common (up to switching) endpoints are called* ***parallel****.*

For any $v \neq s, t$ be $\mathbf{ear}(\boldsymbol{v})$ *the unique number, such that $P^{ear(v)}$ contains v as an internal vertex. By shortcut $\boldsymbol{P_{(v)}}$ we will denote $P^{ear(v)}$.*

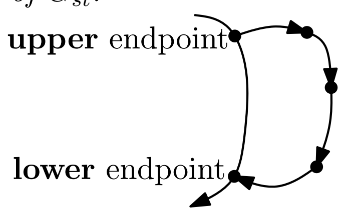
Basic fact about open ear decomposition is that any graph has it if and only if it is biconnected.

Definition 2.3. *Be G biconnected graph, $\{s,t\} \in E(G)$, $(P^0 = \langle s - t\rangle, P^1, \ldots, P^k)$ its open ear decomposition. Let's define graph $\boldsymbol{G}_{\overleftrightarrow{st}}$ to be any orientation of G, such that*

- *$s \rightarrow t$*
- *every ear is oriented in one direction*
- *there is no oriented cycle in the graph G*

Although this definition also gives the notion of G_{ts}, we will be more restrictive and define pair G_{st}, G_{ts} of a graph G such that G_{st} fulfils the previous conditions, and G_{ts} is the inverse of G_{st}, i.e. we get G_{ts} by changing the direction of every edge of G_{st}.

In oriented ear in G_{st} graph, the first endpoint we will denote as ***upper*** *endpoint, the last endpoint we will denote as* ***lower*** *endpoint (in accordance with the pictures).*



Associated st-numbering *of G_{st} graph is such numbering that, $s = 1$, $t = n$ and end-vertex of every edge is bigger than its start-vertex. This numbering fulfills standard definition of* ***st-numbering*** *(for every vertex v, there exists adjacent vertices u, w such that, $u < v < w$). For short we will further use st-numbering always meaning associated st-numbering.*

$\boldsymbol{T_{st}}$ tree of a graph G is a rooted directed tree, that you get by deleting the last edge in every ear of G_{st} except P^0 and rooting it in s.

Note that, by the same definition, we get also T_{ts} tree.

A path from any vertex $v \neq t$ to the root s in the T_{st} tree (denote it $\langle v \longrightarrow s\rangle$) can be constructed inductively: (1) Start in vertex v and (2) from any vertex x which is internal vertex of an ear $P_{(x)}$ continue along the edge of $P_{(x)}$ adjacent to x and oriented towards x (go against to direction of $P_{(x)}$) and reach vertex x' (which by definition has smaller st-number than the x has): $x' < x$.

The step (2) si well defined except the vertex $x = s$ and because of that the st-number strictly decreases, all the vertices of the path are different and the path has to eventually reach the root $s = 1$.

Because T_{st} is a tree, the path is unique.

Definition 2.4. *When $u, v \in G_{st}, v \neq t$ and u lies on the unique path in T_{st} tree from v to s $\langle v \longrightarrow s\rangle$ then by* **st-path** *$\langle v \xrightarrow{\textbf{st}} u\rangle$ we mean the segment of the path $\langle v \longrightarrow s\rangle$ from v to u.*

Lemma 2.5 (st-properties)**.** *For all $R = \langle v \xrightarrow{st} u\rangle$ st-path:*

For all $x, y \neq y' \in R$ such that $R = \langle v \longrightarrow y - y' \longrightarrow x \longrightarrow u\rangle$ holds:

1. *The edge $y - y'$ belongs to the ear $P_{(y)}$.*
2. *$x \leq y' < y$*
3. *$ear(x) \leq ear(y)$*

Proof. 1. and 2.: Follows from the construction of $\langle v \longrightarrow s\rangle$.

3.: By the definition of ear decomposition ear number of endpoint of any ear P^k is less than k. And by the construction of $\langle v \longrightarrow s\rangle$ the only place, where ear number can change is the edge $x - x'$ such that x' is the upper endpoint of an ear $P_{(x)}$. □

Definition 2.6. *In a rooted tree we denote by $\mathsf{lca}(v_1, v_2)$ the least common ancestor of v_1 and v_2, which is a common ancestor (a common point on the unique paths to the root $\langle v_1 \longrightarrow s\rangle, \langle v_2 \longrightarrow s\rangle$) such that every other common ancestor x is an ancestor of the lca ($\langle \mathsf{lca} \xrightarrow{st} x\rangle$). By lca of an ear P we mean lca of its two endpoints.*

There are $\mathsf{FL}^{\mathsf{SL}}$ algorithms for finding a spanning tree in each connected component of a graph G, finding open ear decomposition, orienting ears to have acyclic directed graph G_{st}, associated st-numbering, finding a path from any vertex to root in a rooted tree and counting lca. The reader is referred to [AM04].

Definition 2.7. *A graph is* **planar** *if it can be drawn on the plane so that the edges intersect only at end vertices. Such a drawing is a* **planar drawing***.*

Denote by $E'(G)$ the set of arcs of G: $\{(u, v), (v, u); \{u, v\} \in E(G)\}$.

A **combinatorial embedding** *ϕ of G is a permutation of arcs $\phi : E'(G) \to E'(G)$ such that for any vertex v, ϕ restricted on edges going from v is a cycle.*

Let R maps each arc to its inverse. Then ϕ is a planar combinatorial embedding if and only if the number of orbits f in $(\phi \circ R)$ satisfies Euler's formula $n + f = m + 1 + c$. (Here, n, m, c are the number of vertices, undirected edges, connected components respectively.) For more background, see [Whi84, Section 6-6].

Definition 2.8. *By* ***cyclic ordering*** *at v based on the edge (v, x) denote sequence*

$$\langle (v, x), \phi(v, x), \phi^2(v, x), \ldots, \phi^{-1}(v, x)\rangle$$

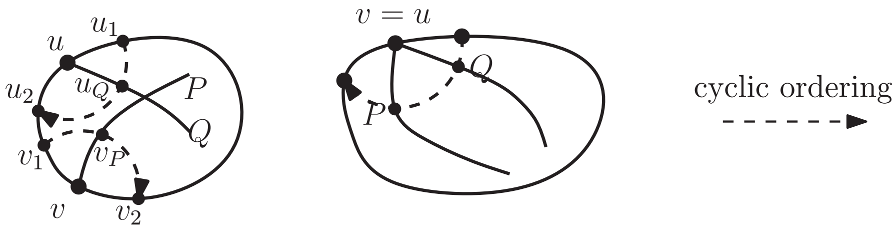


Figure 2.1: Embedding on one side

Let C is a path or a cycle containing vertices u, v. Let there exist paths P and Q edge disjoint with C such that $(u, u_P) \in P$ and $(v, v_P) \in Q$. Then (u, u_P) and (u, u_Q) (also P and Q) are **embedded on one side** *of C means that there exists edges $(u, u_1), (u, u_2)$ in C, $(v, v_1), (v, v_2)$ such that after deleting another edges from cyclic orderings at u and v we get $\langle (v, v_1), (v, v_P), (v, v_2) \rangle$ and $\langle (u, u_1), (u, u_Q), (u, u_2) \rangle$ respectively, where the edges (v, v_1) and (u, u_1) have the same orientation in P.*

Similar definition is in the case, when $u = v$. But we think that figure 2.1 gives a better insight rather than formal definition.

Proposition 2.9 (basic planarity preseving operations)**.** *When a graph G contains vertex v with adjacent edges $e_1, e_2, \ldots, e_n,\ f_1, f_2, \ldots, f_m$, then (i) $\Leftrightarrow$ (ii):*

(i) G is planar and one of its planar embedding ϕ satisfies that there exist permutations i, j such that cyclic ordering at v is $\langle e_{i_1}, \ldots, e_{i_n},\ f_{j_1}, \ldots, f_{j_m} \rangle$

(ii) Make G' from G by replacing v by edge (u, w) and detaching e_x to u, f_y to w (for all x, y). Such G' is planar. (Note that $G = G' \cdot (u, w)$.)

All propositions 2.9, 2.11 and 3.12 in this paper are basic claims about planarity and could be proved by use of the Euler formula, but this is not purpose of this work, thus we left them unproved.

Definition 2.10. *A* **subbridge** *of a subgraph C is a path which intersects with the C in it's two endpoints.*

We premise that most time C will be a cycle.

Proposition 2.11 (basic non-planar embeddings)**.** *When any graph G contains one of the following graphs as a subgraph and has embedding as follows (all explicit descriptions of cyclic orderings are cyclic orderings after deleting edges not contained in the subgraphs) then G is not planar:*

1. *Cycle $C = \langle u \to v \to u \rangle$ and edge disjoint path connected on v and u $P = \langle u \to v \rangle$. First and last edge of P is embedded on the opposite sides of C.*

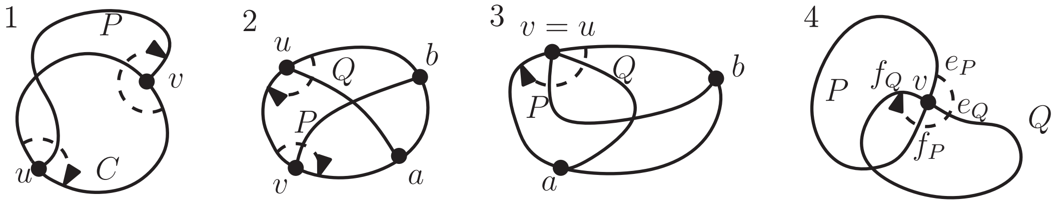


Figure 2.2: Basic non-planar embeddings

2. *A cycle $C = \langle u \to v \to a \to b \to u \rangle$, $P = \langle v \to b \rangle$, $Q = \langle u \to a \rangle$ its disjoint subbridges. Then when P and Q are embedded on one side of C (we say, that subbridges P and Q interlace).*

3. *The same situation as in 2. occurs, when $v = u$ and with extra condition given, that cyclic ordering at v (after deleting another edges) is $\langle \langle v \xrightarrow{C} b \rangle, Q, P, \langle v \xrightarrow{C} a \rangle \rangle$ or $\langle \langle v \xrightarrow{C} b \rangle, \langle v \xrightarrow{C} a \rangle, P, Q \rangle$ (which is stronger than being embedded on one side). We again say, that subbridges P and Q interlace.*

4. *Two cycles P and Q intersect only in vertex v. The ordering at v is $\langle e_P, e_Q, f_P, f_Q \rangle$, where $e_P, f_P \in P$ and $e_Q, f_Q \in Q$.*

Further we will often use another representation of combinatorial embedding ϕ of G_{st}:

Definition 2.12. *For any graph G with G_{st} and its combinatorial embedding ϕ, let us define function $f : E'(G) \to \mathbb{Z}$, such that for any vertex $v \neq s, t$, when $\langle x - v - y \rangle$ is a subpath of $P_{(v)}$, then*

1. $f(v, x) = 0$
2. *For each e edge leaving v:* $e \neq (v, y) \Rightarrow f(\phi(e)) = f(e) + 1$
3. $f(\phi(v, y)) = f(v, y) + 1 - \deg(v)$

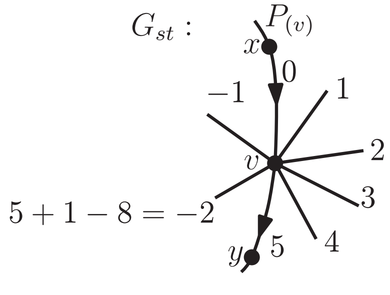


Figure 2.3: Representation of a combinatorial embedding ϕ by a function f

We will again confuse edges and ears ending with them when it is clear which end it is (e.g. by $(f(P_1)$ we mean $f(e_1)))$.

Note also that ears P_1 and P_2 adjacent on v are embedded on one side of $P_{(v)}$ if and only if $\text{sgn}(f(P_1)) = \text{sgn}(f(P_2))$.

Chapter 3

Graph planarity preserving reduction

3.1 Structure of the degree reduction

The degree reduction will consist of five parts. Here is a simple diagram:

$$G_0 \xrightarrow[\text{}]{\overset{1.}{\text{Local replacement}}} G_1 \xrightarrow{\overset{2.}{\text{DSP splitting}}} G_2 \xrightarrow{\overset{3.}{\text{St splitting}}} G_3 \xrightarrow{\overset{4.}{\text{Parallel ears joining}}} G_4 \xrightarrow{\overset{5.}{\text{Parallel ears splitting}}} G_5$$

Every arrow represents an FL algorithm (we will show later) that converts input graph G_i into G_{i+1} such that G_i is planar if and only if G_{i+1} is planar (which we will show later). This gives existence of a single FL algorithm, that converts input graph G_0 into G_5 of maximal degree 3, because the class FL is closed under composition: two subsequent algorithms can be composed into one such way, that in the second algorithm we substitute every attempt to read from the input tape by the whole run of the first algorithm which gives us the desired bit (because it has the tape as its output tape) and for which we need only another logarithmic amount of memory.

Also important is the representation of input and output of every algorithm: the input will always be an open ear decomposition with some $(G_i)_{st}$ orientation which means a list $\langle P_0, \ldots, P^k \rangle$ where P^i is the list of vertices in their G_{st} orientation. This of course does not hold for input graph G_0 hence we have to put in extra preparation phase of this format:

$$G_0 \text{ as an adjacency matrix (e.g.)} \xrightarrow{\mathsf{FL}^{\mathsf{SL}}\text{algorithm}} G_0 \text{ (as a list of oriented ears)} \cdots\blacktriangleright$$

For the third algorithm we also need the st-numbering, hence we need to put in extra preparation phase:

$$\cdots\blacktriangleright G_3 \text{ (as a list of oriented ears)} \xrightarrow{\mathsf{FL}^{\mathsf{SL}}\text{algorithm}} G_3 \text{ in the same format, table of st-numbers} \cdots\blacktriangleright$$

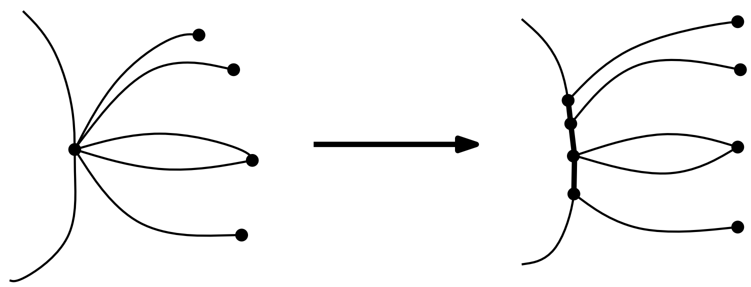
Where arrows represent $\mathsf{FL}^{\mathsf{SL}}$ algorithms. Their existence is proved as we have already noticed in [AM04].

Every algorithm outputs again list of oriented ears except for the fifth algorithm where keeping the ear format would make the algorithm less transparent and, in addition, after that we need no ears any more.

We should make a note about the descriptions of the algorithms: they are slightly inaccurate, because they uses terms like `Replace vertex` v `in a path` P `by a path` $\langle v_{up} \to v \to v_{down} \rangle$. But the LOGSPACE Turing machine does not have a read write tape (except the logarithmic size tape which is too small to contain the description of the whole graph) to perform this operation literally. The command should rather read `Output the list of vertices in ear` P `where` v `will be replaced by the sequence` v_{up},v,v_{down}. Sometimes the correct LOGSPACE implementation is not so straightforward, but it would be much less transparent than our description, but we namely focus on understandability of the algorithms and leave the proper implementation as an easy part of the job to the reader.

Let us compare our approach to the one in [RR94]. There appears a precomputation technique called "local replacement", which has two main stages: the first we used as algorithm 1, the second slightly extended is our algorithm 5.

What do the algorithms do? In general they replace some vertices by connected graphs, in algorithms 1 they are trees, in algorithm 4 edges and in algorithm 5 circles. Algorithms 2 and 3 (which are the main part of our contribution) could be implemented as one, that replaces vertices by paths along the original vertices' ears:



This gives an important semiresult - the graph G_3, where every vertex has at most one ear starting in it (i.e. the vertex has maximal degree three) except some pairs of vertices that can be connected by more parallel ears.

3.2 Local replacement

This technique was introduced earlier, it is described in [RR94].

The purpose of the local replacement is technical: its aim is to avoid this "bad case": an ear P has endpoints u and v and there is a path $\langle u \xrightarrow{\text{st}} v \rangle$ that is disjoint with an ear $P_{(v)}$ (it arrives to the v on another ear). So we want to retach the P (to a new copy of v, call it in our example v_2) such way that new $P_{(v_2)}$ will contain at least the last edge of $\langle u \xrightarrow{\text{st}} v_2 \rangle$. The illustration of such situation is on the figure 3.1

The algorithm goes as follows:

```
For each vertex v
  For each ear P^j with v as endpoint
   Make a copy v_j of v and replace v in P^j by v_j
 End For
End For
Leave ear P^0 = ⟨s − t⟩ unchanged
For each ear P^i for i from 1 to k
  Denote by v upper endpoint, by u lower endpoint
  1) If ear(v) < ear(u)
    Connect u_i to u (add u as a new lower endpoint to P^i)
    Trace the path R := ⟨u -st-> s⟩
    If v ∉ R or the edge before v in R is in P_(v)
     Connect v_i to v (add v as a new upper endpoint to P^i)
    Else be P^j such ear that edge before v in R is in P^j
     Connect v_i to v_j (add v_j as a new upper endpoint to P^i)
  End if
  2) If ear(v) ≥ ear(u)
    Do the same as in 1) with u and v swaped
    and for R use ts-path instead of st-path.
  End if
End for
```

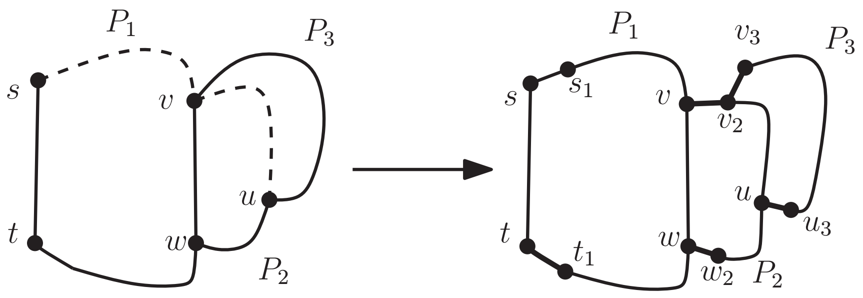


Figure 3.1: Local replacement algorithm

Theorem 3.1. *The resulting graph (call it G_1) is planar, if and only if G is planar. Moreover obtained set of paths forms an open ear decomposition of G_1 in a $(G_1)_{st}$ orientation.*

Proof. This is again discussed in [RR94]. □

For any ear P' of G_1 be P the corresponding ear of G_0 with endpoints u and v, $ear(v) < ear(u)$. If the "bad case" happens, i.e. $\langle u \xrightarrow{\text{st}} v \rangle$ arrives to v on an ear $P^j \neq P_{(v)}$, the algorithm set endpoint of P' to a new vertex v_j on P^j hence now $\langle u \xrightarrow{\text{st}} v_j \rangle'$ arrives to v_j on ear $P'^j = P'_{(v_j)}$. If the bad case does not happen, v remains being endpoint and either at least the last edge of $\langle u \xrightarrow{\text{st}} v \rangle$ is also the last edge of $\langle u \xrightarrow{\text{st}} v \rangle'$, or the st-path does not exist in both of the graphs. Hence the "bad case" is avoided.

3.3 Splitting of oriented and non-oriented ears

You might note that local replacement made some of the degree reduction but most work is still to do.

For further processing we will need an useful tool:

Definition 3.2. *There is* **(Oriented) Double Simple Path** *from v to u ($v \xrightarrow{DSP} u$) if and only if there exists x such that $v \xrightarrow{ts} x$ and $u \xrightarrow{st} x$. The path is $\langle v \xrightarrow{ts} x \xleftarrow{st} u \rangle$.*

Unoriented Double Simple Path *between v and u ($\langle v \xleftrightarrow{DSP} u \rangle$) is either $\langle v \xrightarrow{DSP} u \rangle$ or $\langle u \xrightarrow{DSP} v \rangle$.*

Lemma 3.3 (DSP between ear endpoints)**.** *DSP between two endpoints u and v (without loss of generality $u \xrightarrow{DSP} v$) of an ear P^i can use neither vertex nor edge of the ear.*

Proof. From the definition of open ear decomposition follows that $\text{ear}(u) < i$. By st-properties lemma 2.5 for any vertex $y \in \langle u \xrightarrow{\text{ts}} x \rangle$ $\text{ear}(y) \leq \text{ear}(u) < i$, hence $y \notin P^i$.

The same reasoning works for the st part of the DSP path. □

Lemma 3.4 (DSP-property)**.** *For each path $P = \langle v \xrightarrow{DSP} u \rangle$ and path $\langle w \xrightarrow{st} y \rangle$ such that $y \in P$ there is a path $v \xrightarrow{DSP} w$.*

Proof. By definition of DSP, there exists x such that $P = \langle v \xrightarrow{\text{ts}} x \xleftarrow{\text{st}} u \rangle$. Two cases can occur:
(i) $y \in \langle v \xrightarrow{\text{ts}} x \rangle$ then $\langle v \xrightarrow{\text{ts}} y \xleftarrow{\text{st}} w \rangle$ is DSP.
(ii) $y \in \langle x \xleftarrow{\text{st}} u \rangle$ then $\langle v \xrightarrow{\text{ts}} y \xrightarrow{\text{ts}} x \xleftarrow{\text{st}} w \rangle$ is DSP. □

Since now we will work with the list of st-oriented ears of local replacement graph G_1 obtained by algorithm of 2.3. The algorithm is as follows:

```
For each v (with more ears starting in it)
   Make copies of v v_up and v_down
   Replace v in P_(v) by 3 vertex path ⟨v_up → v → v_down⟩
   For each ear P having v as endpoint
     Denote u the other endpoint of P
     If there is DSP ⟨v ←DSP→ u⟩ then
       If u < v change P endpoint to v_up
       If u > v change P endpoint to v_down
     End If
   End For
End For
```

Let's denote the resulting graph G_2

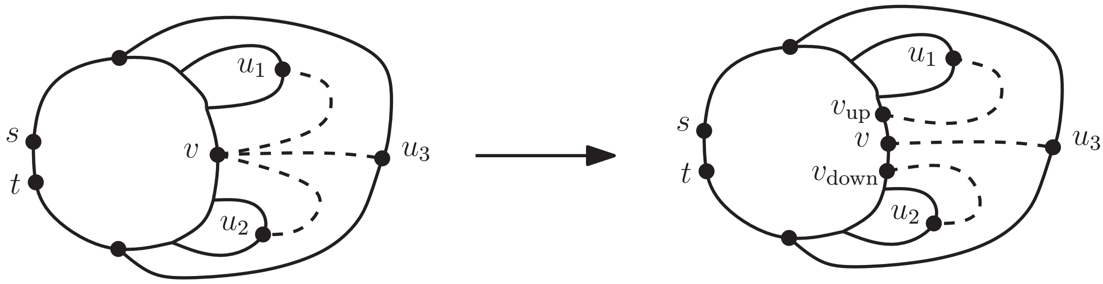


Figure 3.2: DSP splitting algorithm

Theorem 3.5 (DSP-splitting)**.** *G_2 is planar if and only if G_1 is planar.*

Proof. $\Rightarrow$: this part is easy and is common for every proof of this type theorems: since G_1 is obtained from G_2 by collapsing a connected graph (in this particular case $\langle v_{\text{up}} - v - v_{\text{down}} \rangle$) into a vertex which can be done by subsequent contracting of its edges, which by basic planarity preserving operations proposition 2.9 preserves planarity. We get this lemma:

Lemma 3.6 (collapsing of subgraph)**.** *Let G be a planar graph having connected graph T as an induced subgraph and let G' be G with T replaced by a single vertex. Then G' is planar.*

$\Leftarrow$: This part is more difficult but many ideas will be repeated in proofs of following theorems. For arbitrary embedding of G_1 for arbitrary vertex v let's consider a bunch of ears (edges) embedded on one side of $P_{(v)}$. We want to use two times the basic planarity preserving operations lemma 2.9 to know that replacing v by new edge $\langle v_{up} - v \rangle$ and than again by $\langle v - v_{down} \rangle$ does not spoil the planarity. To get assumption of the lemma we show not only there exists the one "special planar embedding ϕ" but that all planar embeddings of G_1 are "special". Hence our goal is to show, that edges belonging to ears with the other endpoint u such that there exists oriented DSP $u \xrightarrow{\text{DSP}} v$ can be planarly embedded only around the edge "above" v. Then edges belonging to ears such that there exists oriented DSP $v \xrightarrow{\text{DSP}} u$ can be planarly embedded only around the edge "below" v. And finally edges belonging to ears with u and v DSP unconnected ($u \overset{\text{DSP}}{\not\leftrightarrow} v$) are in between. Formally using the f representation of ϕ: for all P_1, P_2 with one common endpoint v and others u_1, u_2 respectively, planarly embedded on one side of $P_{(v)}$ (i.e. $\text{sgn}(f(P_1)) = \text{sgn}((f(P_2))))$:

1. If $u_1 \xrightarrow{\text{DSP}} v$ and $v \xrightarrow{\text{DSP}} u_2$ then $|f(P_1)| < |f(P_2)|$.

2. If $u_1 \overset{\text{DSP}}{\not\leftrightarrow} v$ and $v \xrightarrow{\text{DSP}} u_2$ then $|f(P_1)| < |f(P_2)|$.

3. If $u_1 \xrightarrow{\text{DSP}} v$ and $v \overset{\text{DSP}}{\not\leftrightarrow} u_2$ then $|f(P_1)| < |f(P_2)|$.

We will show it indirectly: When P_1 and P_2 are embedded on one side of $P_{(v)}$ and 1., 2. or 3. do not hold, the embedding is not planar:

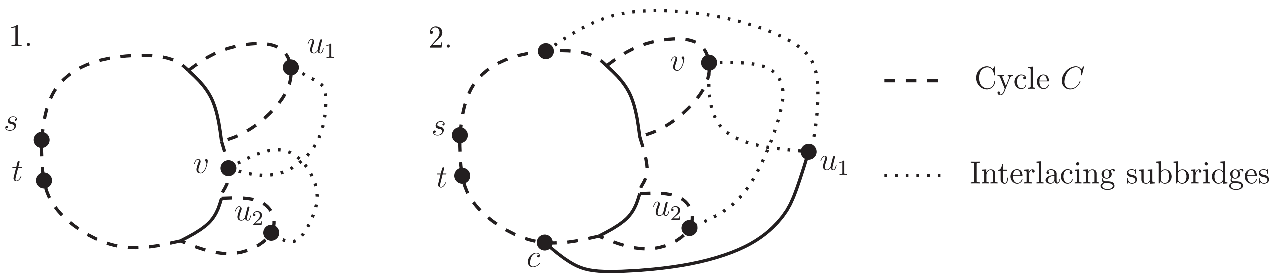


Figure 3.3: DSP splitting correctness

1. P_1 and P_2 are embedded on one side of $P_{(v)}$ and $u_1 \xrightarrow{\text{DSP}} v$ and $v \xrightarrow{\text{DSP}} u_2$ and $|f(P_1)| > |f(P_2)|$: Take cycle $C = \langle s \xleftarrow{\text{st}} u_1 \xrightarrow{\text{DSP}} v \xrightarrow{\text{DSP}} u_2 \xrightarrow{\text{ts}} t - s\rangle$. It has subbridges P_1 and P_2 but they interlace.

   We only need to check, whether embedding of P_1 and P_2 on one side of $P_{(v)}$ implies their embedding on one side of C (we will denote the situation "the sides of $P_{(v)}$ and C at the vertex v correspond"). For which it is sufficient to show that edges in C incident on v are from $P_{(v)}$. Let's discuss the "lower" edge(the first on $\langle v \xrightarrow{\text{DSP}} u_2\rangle$): If the vertex x from the definition of the DSP wasn't v, then ts-path $v \xrightarrow{\text{ts}} x$ would depart v using the $P_{(v)}$ edge by st-properties lemma 2.5-1. If $x = v$ then by local replacement we know, that $\langle u \xrightarrow{\text{st}} v = x\rangle$ arrives to v on an edge from $P_{(v)}$.

   We get this lemma:

   Lemma 3.7 (st-paths after local replacement)**.** *For any $(G_1)_{st}$ with an arbitrary ear P with endpoints u and v and arbitrary vertex x holds:*

   (a) If $x \xleftarrow{st} v$, then the edge by v in $\langle x \xleftarrow{st} v\rangle$ is from $P_{(v)}$. (Of course holds even when v is substituted by u.)

   (b) If $v \xleftarrow{st} u$, then the edge by v in $\langle v \xleftarrow{st} u\rangle$ is from $P_{(v)}$. (This is where the local replacement is needed.)

   (c) (From a and b we get that) when $v \xleftrightarrow{DSP} u$ than the edge by v in $\langle v \xleftrightarrow{DSP} u\rangle$ is also from $P_{(v)}$.

   Note that the lemma also yields that the "upper" v's edge in C is also in P_v.

2. P_1 and P_2 are embedded on one side of $P_{(v)}$ and $u_1 \overset{\text{DSP}}{\not\longleftrightarrow} v$ and $v \xrightarrow{\text{DSP}} u_2$ and $|f(P_1)| > |f(P_2)|$: Take cycle $C = \langle s \xleftarrow{\text{st}} v \xrightarrow{\text{DSP}} u_2 \xrightarrow{\text{ts}} t - s\rangle$. It has a subbridge P_2 and a subbridge $\langle v \xrightarrow{P_1} u_1 \xrightarrow{\text{st}} c\rangle$. Where c is the first vertex on st-path from u_1 belonging to C (There must be always one because there is $\mathsf{lca}(v, u_1)$). Vertex c cannot be from the subpath of C $\langle v \xrightarrow{\text{DSP}} u_2\rangle$, otherwise there was a DSP from v to u_2 (DSP-property lemma 3.4). Moreover P_1 is disjoint with C (DSP between ear endpoints lemma 3.3).

   Hence again we have a cycle C with interlacing subbridges P_1 and $\langle v \xrightarrow{P_1} u_1 \xrightarrow{\text{st}} c\rangle$.

We again did not check that sides of $P_{(v)}$ and C at v correspond, but this is by st-paths after local replacement lemma 3.7.

3. P_1 and P_2 are embedded on one side of $P_{(v)}$ and $u_1 \xrightarrow{\text{DSP}} v$ and $v \not\xleftrightarrow{\text{DSP}} u_2$ and $|f(P_1)| > |f(P_2)|$: it is analogous to 2 (it is sufficient to swap s and t and also P_1 and P_2 and use the proof of 2).

□

Let us note that the local replacement preprocessing is not necessary for the correctness of the DSP splitting (i.e. there is no general planar biconnected graph that would be transformed into non-planar by the DSP splitting) but the proof would be more complicated. In addition, we will inevitably need the argument of the st-paths after local replacement lemma 3.7 in following sections.

3.4 Splitting based on st-numbering

Since now we will work with the list of st-oriented ears of $(G_2)_{st}$ obtained by the DSP splitting algorithm and with a list of st-numbers of all vertices. In this step we will split all non-parallel ears with one common endpoint. The algorithm will be very simple, we will replace each vertex by a bunch of vertices placed along the original vertex's ear, which will be sorted accordingly to st-numbering of non-common vertices:

```
For each v (with more ears starting in it)
   For each u_i, s.t. ∃ ear P: v and u_i are endpoints of P
     Make a copy v_i of v
     For each ear P with endpoints u_i and v
       Replace the vertex v in P by v_i
     End For
   End For
   If ⟨v <-DSP-> u_i⟩ for all i then //type v_up or v_down
     Sort ⟨v_i⟩_i vertices accordingly to −st-number(u_i)
   Else //type v
     Sort ⟨v_i⟩_i vertices accordingly to st-numbers(u_i)
   Replace v in P_(v) by a path of sorted v_i
End For
```

Let's denote the resulting graph G_3

Theorem 3.8 (st-splitting). *G_3 is planar if and only if G_2 is planar.*

Proof. $\Rightarrow$: Again by collapsing of subgraph lemma 3.6.

$\Leftarrow$: We want to show, that some orderings around any vertex in G_2 are not possible when embedding is planar. This will be done by considering arbitrary ears P_1, P_2 with one common endpoint v and u_1 and u_2 as other endpoints: without loss of generality $u_1 < u_2$.

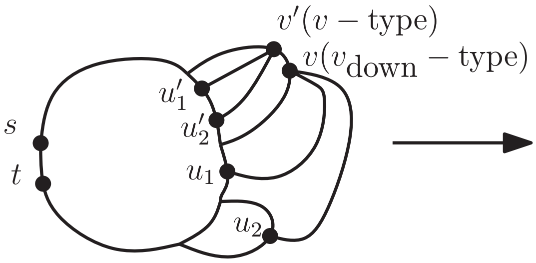


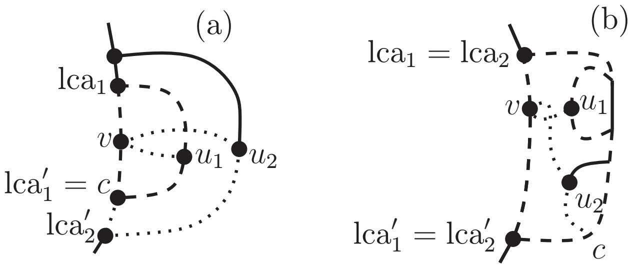
Figure 3.4: St splitting algorithm

Note that DSP-splitting algorithm made in graph G_2 three type of vertices (not mentioning that of degree less or equal 3). Hence v can be:
(i) v_{down}-type: $\Rightarrow v \xrightarrow{\text{DSP}} u_1$ and $v \xrightarrow{\text{DSP}} u_2$ (case 2).
(ii) v_{up}-type: $\Rightarrow u_1 \xrightarrow{\text{DSP}} v$ and $u_2 \xrightarrow{\text{DSP}} v$. We won't handle this case separately, because it can be reduced to case 2 by swapping vertices s and t).
(iii) v-type: $\Rightarrow v \overset{\text{DSP}}{\not\longleftrightarrow} u_1$ and $v \overset{\text{DSP}}{\not\longleftrightarrow} u_2$ (case 1).

Given that P_1 and P_2 are embedded on one side of $P_{(v)}$, we would like to show that when case 1 occurs and $|f(P_1)| > |f(P_2)|$, embedding is not planar. In case 2 the same for $|f(P_1)| < |f(P_2)|$.

For the purposes of the proof let's denote $\mathsf{lca}_i = \mathsf{lca}(v, u_i)$ in T_{st} and $\mathsf{lca}'_i = \mathsf{lca}(v, u_i)$ in T_{ts} tree for $i \in \{1, 2\}$.

1. $v \overset{\text{DSP}}{\not\longleftrightarrow} u_1$ and $v \overset{\text{DSP}}{\not\longleftrightarrow} u_2$ and $|f(P_1)| > |f(P_2)|$:



We will illustrate two subcases: (a) $u_1 \overset{\text{DSP}}{\not\longleftrightarrow} u_2$ and (b) $u_1 \overset{\text{DSP}}{\longleftrightarrow} u_2$. For the subcase (a) it even holds (and the proof could be easily extended to show) that when P_1 and P_2 are embedded on one side of $P_{(v)}$, the embedding is not planar at all. But this is more than we actually need to proof.

Figure 3.5: st splitting correctness - case 1

Take the cycle $C = \langle v \xrightarrow{\text{st}} \mathsf{lca}_1 \xleftarrow{\text{st}} u_1 \xrightarrow{\text{ts}} \mathsf{lca}'_1 \xleftarrow{\text{ts}} v \rangle$. One subbridge is P_1. The other is $R = \langle v \xrightarrow{P_2} u_2 \xleftrightarrow{\text{ts}} c \rangle$, where $c \in C$ is the nearest vertex to u_2 along the path $\langle u_2 \xrightarrow{\text{ts}} \mathsf{lca}'_2 \xleftrightarrow{\text{ts}} \mathsf{lca}'_1 \rangle$ (possibly $c = u_2$. By $\xleftrightarrow{\text{ts}}$ we mean arbitrary path in T_{ts} tree.

The only nontrivial question about disjointness is disjointness of P_1 and $R' = \langle u_2 \xrightarrow{\text{ts}} \mathsf{lca}'_2 \rangle$. But if R' entered P_2, it would follow it until reach u_1 (contradiction with $u_2 > u_1$) or reach v (contradiction with $v \overset{\text{DSP}}{\not\longleftrightarrow} u_2$).

Also note that all lcas have to be nonequal to v, otherwise there would be $v \xleftrightarrow{\text{DSP}} u_1$ or $v \xleftrightarrow{\text{DSP}} u_2$. Hence we get interlacing subbridges P_1 and R.

2. $v \overset{\text{DSP}}{\longleftrightarrow} u_1$ and $v \overset{\text{DSP}}{\longleftrightarrow} u_2$ (without loss of generality $v \overset{\text{DSP}}{\longrightarrow} u_1$ and $v \overset{\text{DSP}}{\longrightarrow} u_2$) and $|f(P_1)| < |f(P_2)|$:

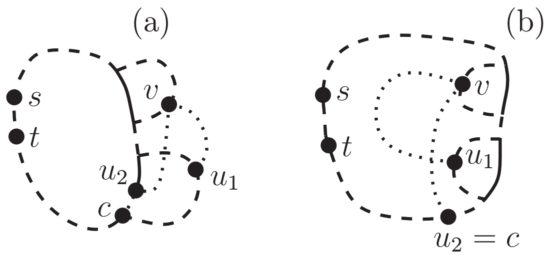


Figure 3.6: st splitting correctness - case 2

Let's take a cycle $C = \langle s \overset{\text{st}}{\longleftarrow} v \overset{\text{DSP}}{\longrightarrow} u_1 \overset{\text{ts}}{\longrightarrow} t - s \rangle$. One subbridge is P_1. The other is more complicated. It is a path $R = \langle v \overset{P_2}{\longrightarrow} u_2 \overset{\text{ts}}{\longrightarrow} c \rangle$, where c is the first vertex on the ts-path from u_2 belonging to C (possibly $c = u_2$). It holds, that $c \geq u_2 > u_1 \Rightarrow c > u_1$. Subbridge P_1 and R interlaces.

We need to check, that sides of C at v corresponds with sides of $P_{(v)}$. but this holds by DSP after local replacement lemma 3.7.

Now remains disjointness of paths. We need to check part of C $\langle v \overset{\text{DSP}}{\longrightarrow} u_1 \rangle$ versus P_2. By definition $\langle v \overset{\text{DSP}}{\longrightarrow} u_1 \rangle = \langle v \overset{\text{ts}}{\longrightarrow} x \overset{\text{st}}{\longleftarrow} u_1 \rangle$ but because $x \neq v$ (by st-paths after local replacement lemma 3.7), the $\langle x \overset{\text{st}}{\longleftarrow} u_1 \rangle$ cannot enter P_2, because otherwise it would follow P_2 until reach v hence again it would $x = v$. (The ts-part cannot enter P_2 because ear numbers decreases by st-properties lemma 3.) Hence $\langle v \overset{\text{DSP}}{\longrightarrow} u_1 \rangle$ is disjoint with P_2.

□

3.5 Splitting of parallel ears

Since now we will work with list of oriented ears of G_3, we obtained from the st splitting algorithm. In this easy preparation step we will make all parallel ears sit on a common ear.

The algorithm follows:

```
For all maximal sets of parallel ears 𝒫 = {P_1, ..., P_j}
  (denote their endpoints u and v)
  If u and v don't lie on a common ear
   Make copies of u and v named u' and v'.
   Add u' and v' to P_1 as new endpoints (u and v become internal)
   Replace u and v in P_(u) and P_(v) by u' and v'
  End If
```

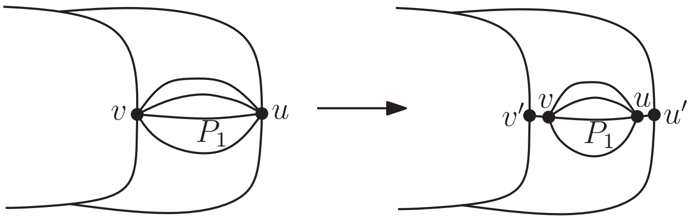


Figure 3.7: Parallel ears joining algorithm

```
End For
```

Let's denote the resulting graph G_4.

Theorem 3.9 (parallel ears joining)**.** *G_3 is planar, if and only if G_4 is planar.*

Proof. Again, non trivial part is "⇒": Suppose for contradiction that, there is a planar embedding of G_3 which cannot be (by basic planarity preserving operations lemma 2.9) transformed into embedding of G_4, which means there are parallel ears P_1 and P_2 in G_3 with common endpoints u and v that are embedded on opposite sides of (without loss of generality) $P_{(v)}$. Two cases follows:

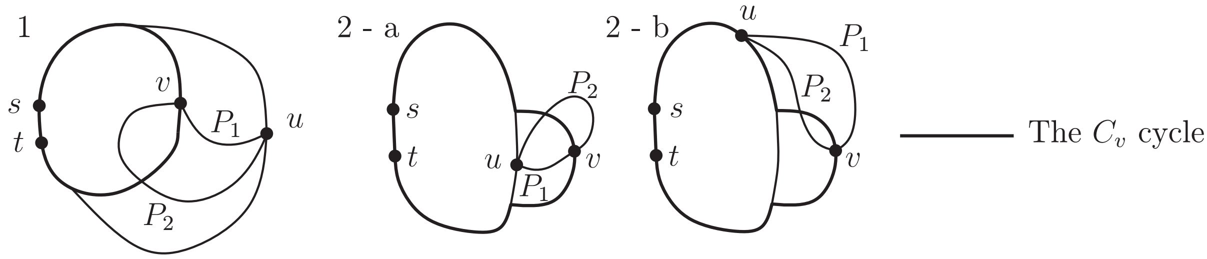


Figure 3.8: Parallel ears joining correctness

1. $P_{(v)}$ has lower ear-number than $P_{(u)}$, then we can consider cycles $P = \langle v \xrightarrow{P_1} u \xleftarrow{P_2} v\rangle$ and $C_v = \langle v \xrightarrow{\text{st}} s - t \xleftarrow{\text{ts}} v\rangle$. Then v is the only point in their intersection (all other vertices on C_v lies on an ear with less or equal ear-number than $P_{(v)}$ has and all other vertices on P has bigger ear number). Hence P and C_V are interlacing cycles, which is a contradiction (by basic non-planar embeddings proposition 2.11).

2. $P_{(v)}$ has greater ear-number than $P_{(u)}$. Now if u does not lie on C_v (case a), the previous reasoning leads to contradiction. If does (case b),

   (i) we still know by previous reasoning, that P_1 and P_2 are embedded on one side of C_u.

(ii) But by st-paths after local replacement lemma 3.7 we know that sides of C_u and C_v at u correspond ($C_u \cap C_v$ contains not only u but also its two neighbors (in C_u)).

Thus (i) together with (ii) gives that P_1 and P_2 are at the vertex u embedded on one side of C_v. And it gives another non-planar configuration from basic non-planar embeddings proposition 2.11.

□

The very last step deals with all parallel ears of the G_4. Here we need to have s and t of maximal degree 2, which can be achieved by making two auxiliary vertices on (old) (s,t) edge and giving them names new-s and new-t.

For the purposes of this last section let us define set of symbols $[n] = \{r, 0, 1, ..., n\}$.

Parallel ears splitting algorithm follows:

```
For all maximal sets of parallel ears 𝒫 = {P_1, P_2, ..., P_n}
  Denote their endpoints u and v
  Denote P_0 = ⟨v <-P(v)-> u⟩ //Because of the previous algorithm P_(v) = P_(u)
  Denote P_r = ⟨v -st-> s − t <-ts- u⟩

  1. Make new vertices v_r, v_0, v_1, ..., v_n
     For all ears P ∉ 𝒫 with endpoints p and q
       Trace vertices x ∈ ⟨p -st-> s⟩ from x = p in the path order
        until there is a such that x ∈ P_a //(there is at least x = s ∈ P_r)
       Trace vertices x ∈ ⟨q -st-> s⟩ from x = q in the path order
        until there is b such that x ∈ P_b //(there is at least x = s ∈ P_r)
       If a ≠ b
        Connect v_a and v_b
       End If
     End For
     Denote the constructed subgraph V
  2. If V has vertex of degree 3 or more
       Reject
     If V has a cycle containing not all vertices {v_i, i ∈ [n]}
       Reject
     //If we did not reject, V is consisting of disjoint paths
     //or one cycle
  3. Make V̄ := V + arbitrary matching s.t.  V̄ is cycle
     //i.e.  arbitrarily join the paths of V into one cycle
  4. Make new vertices u_r, u_0, u_1, ..., u_n
     Form with them a cycle the same way as v_r, ..., v_n do
  5. "Replace" v and u in P_r, P_0, P_1, ..., P_n by corresponding copies

End For
```

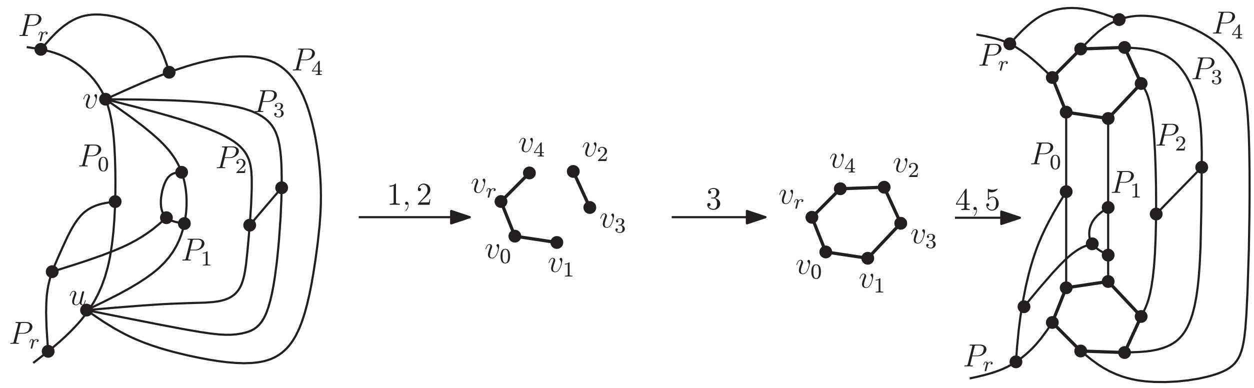


Figure 3.9: Parallel ears splitting algorithm

First consider what are the parts $P_r, P_0, P_1, \dots, P_n$. For the definition of P_0 as a part of common ear of u and v we get necessity of parallel ear joining. Also note that we assured before, that degree of s and t is at most three, hence u and v cannot be s and t and thus is $P_r \neq P_0$. That's why $P_i, i \in [n]$ are well defined disjoint subbridges of $C = \{u, v\}$ in G. The goal of the algorithm is to find paths between them disjoint with them. We just don't need to find all paths, we need to find for every subbridge any, if there are some connecting it to the rest of the world. More exactly:

Lemma 3.10 (connectivity detecting). *For all $i, j \in [n]$, if there is a path O disjoint with defined subbridges connecting P_i and P_j, then there exist $k_0, ..., k_x \in [n]$, s.t. $i = k_0$, $j = k_x$ and for all $\beta \in \{0, .., x-1\}$ v_{k_β} is connected by the algorithm with $v_{k_{\beta+1}}$.*

Proof. Every edge of path O belongs to some ear. Hence we can take sequence of ears $P^1, P^2, \dots, P^y$ whose edges are visited by O (in this order, indeces do not mean ear number). For every pair of consequent ears $P^\alpha, P^{\alpha+1}$ define $I_\alpha^{\alpha+1}$ to be st-path starting from the vertex, where P^α and $P^{\alpha+1}$ meet (denote it w_a^{a+1}) going to the root s (thus $I_\alpha^{\alpha+1} = \langle w_\alpha^{\alpha+1} \xrightarrow{\text{st}} s \rangle$). Hence we have sequence $I_0^1, I_1^2, \dots, I_y^{y+1}$ (we just need to extend the definition by setting $P^0 := P_i$ and $P^{y+1} := P_j$).

Let us examine these paths in the same way the algorithm did with the st-paths starting from endpoints: define a mapping $\omega : \{\text{st-paths}\} \to [n]$ by setting for any vertex w $\omega(\langle w \xrightarrow{\text{st}} s \rangle) := c$, where $c \in [n]$ and P_c has the intersection with $\langle w \xrightarrow{\text{st}} s \rangle$ nearest to the w among all ears $P_r, P_0, \dots, P_n$. Thus you can notice that computations of ω are done in the step 1.: $a := \omega(\langle p \xrightarrow{\text{st}} s \rangle)$ and $b := \omega(\langle q \xrightarrow{\text{st}} s \rangle)$.

Let's have a look at the sequence $\omega(I_0^1), \omega(I_1^2), \dots, \omega(I_y^{y+1})$. It starts with i and ends with j. Now for every pair $I_{\alpha-1}^\alpha, I_\alpha^{\alpha+1}$ $\omega(I_{\alpha-1}^\alpha) \neq \omega(I_\alpha^{\alpha+1})$ yields that one of the $w_{\alpha-1}^\alpha$ and $w_\alpha^{\alpha+1}$ which both lies in P^α has to be the lower endpoint of P^α. Otherwise both st-paths $I_{\alpha-1}^\alpha, I_\alpha^{\alpha+1}$ would join even before leaving the ear P^α and hence their ω property would be the same. But this yields that the step 1. after examining the ear P^α connects $v_{\omega(I_{\alpha-1}^\alpha)}$ and $v_{\omega(I_\alpha^{\alpha+1})}$.

Hence the desired sequence $(k_\beta)_{\beta=0}^x$ is obtained from sequence $(\omega(I_\alpha^{\alpha+1}))_{\alpha=0}^y$ by replacing every maximal subsequence of the same letter by the single letter. □

Note, that the proof could have been done by induction on the length of the path O using the same central argument and less words, but we think this way gives a better insight into the problem.

Corollary 3.11 (components of connectivity)**.** *Let the graph V has components of connectivity $V_1, \dots, V_m$. For any $l \in 1, \dots, m$ and V_l let us define $A_l \subseteq [n]$ such that $A_l = \{a; v_a \in V_l\}$. Then $G_4 \setminus \{u, v\}$ consists of components of connectivity $G^1, \dots, G^m$ and for all $l \in \{1, \dots, m\}$ G^l contains all ears $P_a, a \in A_l$.*

Proof. Every path in $G_4 \setminus \{u, v\}$ can be decomposed into parts disjoint with the defined subbridges and for every such part, we have path in V as the connectivity detecting lemma 3.10 shows. And trivially for every edge (and hence for every path) in V we have corresponding path in $G \setminus \{u, v\}$. Hence we get the desired correspondence of components of connectivity between V and $G_4 \setminus \{v, u\}$. □

Note that after the step 2. of the algorithm if we did not reject, the components of connectivity (when there are more than one) must be all paths. Hence for all $l \in \{1, \dots, m\}$ there exists a labelling of elements of A_l: $A_l = \{a_1, \dots, a_z\}$ such that $V_l = \langle v_{a_1} - v_{a_2} - \dots - v_{a_z} \rangle$.

We can consider the parallel ears splitting algorithm to be composition of two sequential steps. The first would be this:

Proposition 3.12 (triconnectivity sticking)**.** *For each planar graph G and for all $u, v \in V(G)$, if $G \setminus \{u, v\}$ has components of connectivity $G^1, \dots, G^m$ then following graph $G^{u,v}$ is planar ($\dot\cup$ is disjoint union):*

$$V(G^{u,v}) := (V(G) \setminus \{u, v\}) \,\dot\cup \{v^1, \dots, v^m, u^1, \dots, u^m\}$$

$$E(G^{u,v}) := E(G)|V(G^{u,v}) \cup \{(v^l, x); (v, x) \in E(G) \wedge x \in G^l\} \cup \{(v^1, v^2), \dots, (v^m, v^1)\} \cup$$
$$\cup \{(u^l, x); (u, x) \in E(G) \wedge x \in G^l\} \cup \{(u^1, u^2), \dots, (u^m, u^1)\}$$

Note that $G^{u,v}$ is planar for any numbering of components of connectivity, hence we can often choose more ways in the step 3. of the algorithm and we will always be successful.

Now we will show that when two subbridges P_i and P_j are connected by a path disjoint with subbridges in $\mathcal{P}$, it makes them strongly dependent in the planar embedding:

Lemma 3.13 (ear adjacency)**.** *Take assumptions of the main step of the algorithm. When subbridges P_i and P_j are connected by a path O disjoint with all subbridges in $\mathcal{P}$ (specially when the algorithm connects v_i and v_j), then in any planar embedding of G_4 P_i and P_j are adjacent in cyclic ordering around both v and u. Formally $\phi(v, P_i) = P_j$ or $\phi(v, P_j) = P_i$. Same holds, when v is replaced by u.*

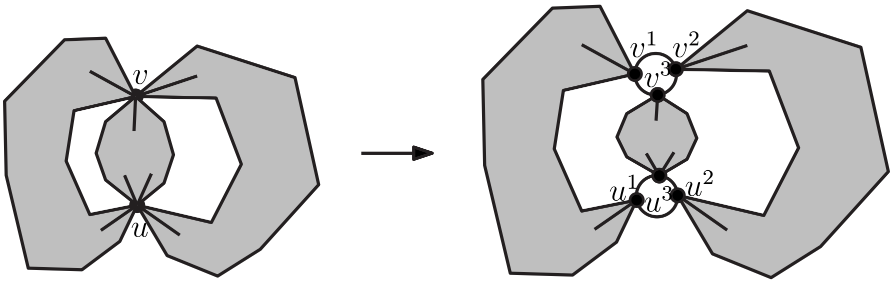


Figure 3.10: Triconnectivity sticking

Proof. Subbridges P_i and P_j form a cycle, denote it C. O is its subbridge. Denote inside of C to be the area, where O is embedded. For contradiction suppose there is an ear $P \in \mathcal{P}$ that is embedded also inside the C. But O is connected on C at internal vertices of P_i and P_j, whereas P is connected on C at u and v. Hence P and O interlaces. This yields contradiction. □

Corollary 3.14 (rejecting)**.** *When algorithm rejects, G_4 is not planar.*

Proof. When algorithm rejects at first condition of step 2., there is a vertex v in a V of degree 3 or more. By ear adjacency lemma 3.13 it means that there must be three adjacent subbridges to the v's subbridge. But in the plane this is impossible. Formally exist P, P_i, P_j, P_k with endpoints $u, v : \phi(P_i) = P$ and $\phi(P) = P_j$, but then $\phi(P) = P_k$ or $\phi(P_k) = P$ cannot occur.

When algorithm rejects at the second condition of step 2., there is a cycle in V containing not all v's vertices, but by the ear adjacency lemma 3.13 there must be the same cycle in ϕ, but then it is not proper cyclic ordering around the vertex v (and also u). □

Theorem 3.15 (final)**.** *G_4 is planar, if and only if algorithm does not reject and G_5 is planar.*

Proof. $\Leftarrow$: This is again trivial, because G_5 is G_4 with some vertices replaced by connected graphs (in this case by the cycles).

$\Rightarrow$: Rejecting lemma 3.14 gives the first half of the implication.

Because of the triconnectivity sticking proposition 3.12, we have to prove that if $G_4^{u,v}$ (v and u replaced by a cycles of $v^1, \dots, v^m$ and $u^1, \dots u^m$) is planar then G_5 is planar. This will be proved by basic planarity preserving operations 2.9 if we show, that there is a planar embedding of $G_4^{u,v}$ ϕ: for all $l \in \{1, \dots, m\}$ and for corresponding path $V_l = \langle v_{a_1} - v_{a_2} - \dots - v_{a_z} \rangle$ holds that $\phi(v^l, P_{a_\gamma}) = P_{a_{\gamma+1}}$ and $\phi(u^l, P_{a_{\gamma+1}}) = P_{a_\gamma}$, for all $\gamma = 1, \dots, z-1$.

By the assumption that $G_4^{u,v}$ is planar and by ear adjacency lemma 3.13 we know, that there is a planar embedding ϕ' : for all $\gamma = 1, \dots z-1 : \phi'(v, P_{a_\gamma}) = P_{a_{\gamma+1}}$ (case a) or for all $\gamma = 1, \dots z-1 : \phi'(v, P_{a_{\gamma+1}}) = P_{a_\gamma}$ (case b). In case a) set $\phi := \phi'$. In case b) set $\phi := \phi'^{-1}$ (It is also planar, because we get the same faces as before, we can imagine it as mirroring of the drawing.)

The very last step is to prove, that now $\phi(u^l, P_{a_{\gamma+1}}) = P_{a_\gamma}$ for all $\gamma = 1, \ldots, z-1$. By contradiction for all $\gamma = 1, \ldots, z-1$ $\phi(u^l, P_{a_\gamma}) = P_{a_{\gamma+1}}$ (by ear adjacency lemma 3.13 there is no other possibility), hence also $\phi(u^l, P_{a_{z-1}}) = P_{a_z}$ and $\phi(u^l, P_{a_z}) = P$ where P is a path from u^l to v^l disjoint with both $P_{a_{z-1}}$ and $P_{a_{z+1}}$. But P plus $P_{a_{z-1}}$ forms a cycle and P_{a_z} is a a subbridge of the cycle embedded on both sides of the cycle: contradiction.

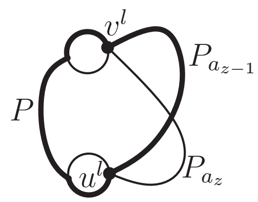


Figure 3.11: Final theorem

□

Chapter 4

Conclusion

So we have proven that the planarity testing of maximal degree three graphs can be extended to solve the general case. The main motivation for investigating this was to come up with an algorithm focused on space complexity rather than on practical effectiveness. This proof is still rather difficult to give a talk about it but it is probably due to the difficulty of the planarity problem.

There is another approach probably still left unaddressed (and is mentioned in [AM04]): to "tune" the parallel algorithm in [RR94] for the purposes of the space complexity.

Bibliography

[AM04] Eric Allender and Meena Mahajan. The complexity of planarity testing. *Inform. and Comput.*, 189(1):117–134, 2004.

[Rei84] John H. Reif. Symmetric complementation. *J. Assoc. Comput. Mach.*, 31(2):401–421, 1984.

[Rei05] Omer Reingold. Undirected ST-connectivity in log-space. In *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 376–385, New York, 2005. ACM.

[RR94] Vijaya Ramachandran and John Reif. Planarity testing in parallel. *J. Comput. System Sci.*, 49(3):517–561, 1994.

[Sip96] Michael Sipser. *Introduction to the Theory of Computation.* International Thomson Publishing, 1996.

[Whi84] Arthur T. White. *Graphs, groups and surfaces*, volume 8 of *North-Holland Mathematics Studies.* North-Holland Publishing Co., Amsterdam, second edition, 1984.