Computing the stretch of an embedded graph

Sergio Cabello*[1], Markus Chimani†[2], and Petr Hliněný‡[3]

[1]Department of Mathematics, FMF, University of Ljubljana, Slovenia
[2]Theoretical Computer Science, Department of Maths/CS, Osnabrück University, Germany
[3]Faculty of Informatics, Masaryk University, Brno, Czech Republic

Abstract

Let G be a graph embedded in an orientable surface Σ, possibly with edge weights, and denote by $\text{len}(\gamma)$ the length (the number of edges or the sum of the edge weights) of a cycle γ in G. The stretch of a graph embedded on a surface is the minimum of $\text{len}(\alpha)\cdot\text{len}(\beta)$ over all pairs of cycles α and β that cross exactly once. We provide an algorithm to compute the stretch of an embedded graph in time $O(g^4 n\log n)$ with high probability, or in time $O(g^4 n\log^2 n)$ in the worst case, where g is the genus of the surface Σ and n is the number of vertices in G.

Introduction

Consider a graph G embedded on an orientable surface Σ of genus g. What can be said about the crossing number of G in the plane? Is it computable in polynomial time? If it is not, can we obtain a reasonable approximation in polynomial time? Unfortunately, Cabello and Mohar [3] show that the crossing number of such graphs is not computable in polynomial time, even when Σ is the torus. Djidjev and Vrt'o [4] show that the crossing number of G is upper bounded by $O(g\Delta n)$, where n is the number of vertices in G and Δ is the maximum degree of G. This is an improvement over the previous bound of $O(C^g\Delta n)$, for some constant C, by Börözky, Pach and Tóth [1]. Under some mild assumptions about the density of the embedding of G, Hliněný and Chimani [5] give a $(3\cdot 2^{3g+2}\Delta^2)$-approximation algorithm for the crossing number of G. This last work is the main motivation for our research.

Hliněný and Chimani [5] define the ***stretch*** of an embedded graph G as

$$\texttt{str}(G) \;=\; \min_{\alpha,\beta}\{\text{len}(\alpha)\cdot\text{len}(\beta)\},$$

where α,β ranges over all pairs of cycles in G that cross exactly once. Here, $\text{len}(\alpha)$ denotes the number of edges in α and a ***cycle*** is a closed walk in a graph without repeated vertices. A precise definition of what "crossing exactly once" means is given in Section 1.2. The stretch plays a fundamental role in their analysis of the algorithm. The concept of stretch can be generalized to the case of positive edge-weighted graphs in a natural way: take $\text{len}(\alpha)$ to be the sum of the edge-weights along the cycle α. Henceforth we will assume this more general definition of stretch.

It is worth noting that, if two cycles α and β are crossing once, then they must be both (surface-)non-separating. That is, cutting the surface along α or β does not disconnect the surface. This is so because any cycle crosses a (surface-)separating cycle an even number of times. Thus, when computing the stretch, we can restrict our attention to pairs (α,β) of non-separating cycles.

In this paper we provide an algorithm to compute the stretch of an embedded graph in time $O(g^4 n\log n)$ with high probability, or in time $O(g^4 n\log^2 n)$ in the worst case, where g is the genus of the surface Σ and n is the number of vertices in G.

Overview of the approach. Let us provide an informal overview of the main ideas. We show the following recursive property of the stretch: it is defined either by the shortest non-separating cycle α^* and one other cycle crossing α^* exactly once, or by the stretch of surface obtained by cutting along α^* and pasting disks. We do not prove this claim directly, but using a detour through another concept: odd-stretch.

The definition of odd-stretch resembles the definition of stretch, but we allow closed walks α and β, instead of just cycles, and allow an odd number of crossings, instead of exactly one crossing. It turns out that the stretch and odd-stretch of a graph is the same. However, working with the odd-stretch is easier because we only need to take care of the parity of crossings and, when constructing new closed walks

*Supported by the Slovenian Research Agency, program P1-0297, project J1-4106, and within the EUROCORES Programme EUROGIGA (project GReGAS) of the European Science Foundation.

†This research was conducted while being funded by a Carl-Zeiss-Foundation juniorprofessorship, and partially supported by EUROCORES Programme EUROGIGA (project GraDR) of the European Science Foundation.

‡Supported by the Czech Science Foundation, EUROCORES grant GIG/11/E023 (project GraDR).

via exchange arguments, we do not need to take care to construct cycles. Finally, we prove the aforementioned recursive property for the odd-stretch factor.

The eventual algorithm, given in Figure 1 is very simple. However, there is a fine point we have to take care of to obtain a polynomial-time algorithm. Repeatedly cutting along shortest non-separating cycles and pasting disks may give rise to an exponential growth in the size of the graphs: at each cut we make copies of the vertices along the cycle and thus the number of vertices may nearly double at each iteration. However, if at some iteration we get a shortest non-separating cycle with more vertices than the original graph, we can finish the recursive search. In this way we avoid the potentially exponential growth in the size of the graphs.

1 Odd-stretch

In this section we introduce the concept of odd-stretch, which is a generalization of stretch. We first discuss crossings for curves in general position and then crossings for closed walks in a graph. Finally, we define the odd-stretch, discuss some of its properties and, eventually, show that the odd-stretch is the same as the stretch.

1.1 Crossings of curves in general position

Two curves C and C' on a surface Σ are in ***general position*** if they have a finite number of intersections and, at each intersection, they cross transversally. For two curves $C = C(t)$ and $C' = C'(t)$ in general position, the ***set of crossings*** is

$$X(C, C') \;=\; \{x \in \Sigma \mid \exists t, t' \text{ s.t. } x = C(t) = C'(t')\}.$$

Two curves C and C' in general position ***cross*** k times if and only if $k = |X(C, C')|$. We will use the following (intuitive) fact: the number of crossings between two closed curves, modulo 2, is invariant under small perturbations of any of the two closed curves.

1.2 Crossings of closed walks

Two closed walks α and β in G cross k times if and only if: there are arbitrarily small perturbations of α and β to general position that cross k times, and any small enough perturbation of α and β has at least k intersecting points. Moreover, we can always assume that the crossings of the perturbations occur in a neighborhood of the vertices. We denote by $\texttt{cr}(\alpha, \beta)$ the number of crossings between α and β.

For any closed walk α, the set of closed walks in G that cross α an odd number of times satisfies the so-called 3-path condition. The next lemma states this in an equivalent way for easier use later on. This property was already noted in [5].

Lemma 1 *Let α be a closed walk and let γ be a closed walk crossing α an odd number of times. Let x and y be two vertices on γ and let π be some walk from x to y. Let γ' be the closed walk defined by concatenating $\gamma[y \to x]$ and π. Let γ'' be the closed walk defined by concatenating $\gamma[x \to y]$ and the reverse of π. Then either γ' or γ'' cross α an odd number of times.*

1.3 Odd-stretch of an embedded graph

The odd-stretch of an embedded graph G is

$$\texttt{oddstr}(G) \;=\; \min_{\alpha,\beta}\{\text{len}(\alpha) \cdot \text{len}(\beta)\},$$

where α, β ranges over all pairs of closed walks in G that cross an odd number of times. We next remark the two main differences with the previous stretch: α and β iterate over closed walks, instead of cycles, and the curves can cross an odd number of times, instead of exactly once. A priori, the stretch and the odd-stretch of an embedded graph are different. A posteriori we can see that they are the same; this was already noted in [5].

Lemma 2 *The odd-stretch of G and the stretch of G are the same.*

2 Algorithm for computing the stretch factor

We will use the following two properties:

Lemma 3 ([2]) *Let G be a graph with m vertices embedded in a surface of genus g.*

- *We can compute a shortest non-separating cycle in time $O(g^2 m \log m)$ with high probability, or in time $O(g^2 m \log^2 m)$ in the worst case.*
- *For any given non-separating cycle α, we can compute a shortest cycle of G that crosses α exactly once in time $O(gm \log m)$ with high probability, or in time $O(gm \log^2 m)$ in the worst case.*

Lemma 4 *Let α be a shortest non-separating cycle in G. For any two vertices x and y on α, α contains a shortest path from x to y or from y and x.*

The algorithm for computing the stretch of an embedded graph is given in Figure 1. We first discuss its time complexity and then its correctness.

Lemma 5 *Algorithm* ComputeStretch *has time complexity $O(g^4 n \log n)$ with high probability, or $O(g^4 n \log^2 n)$ in the worst case, where n is the number of vertices in G.*

```
Algorithm ComputeStretch
Input: graph $G$ embedded in surface $\Sigma$
Output: stretch of $G$
1.  $i \leftarrow 1$;
2.  $(G_1, \Sigma_1) \leftarrow (G, \Sigma)$;
3.  str $\leftarrow \infty$;
4.  while $\Sigma_i$ not the sphere and $|V(G_i)| \leq g \cdot |V(G)|$ do
5.          $\alpha_i \leftarrow$ shortest non-separating cycle in $G_i$;
6.          $\beta_i \leftarrow$ shortest cycle crossing $\alpha_i$ exactly once;
7.          str $\leftarrow \min\{\text{str}, \text{len}(\alpha_i) \cdot \text{len}(\beta_i)\}$;
8.          $(G_{i+1}, \Sigma_{i+1}) \leftarrow$ cut $(G_i, \Sigma_i)$ along $\alpha_i$ and attach disks to the boundaries;
9.          $i \leftarrow i + 1$;
10. return str
```

Figure 1: Algorithm ComputeStretch to compute the stretch of an embedded graph.

Proof. Because of Lemma 3, in each iteration of the while loop we need $O(g_i^2 n_i \log n_i)$ time whp, or $O(g_i^2 n_i \log^2 n_i)$ in the worst case, where n_i is the number of vertices in G_i and g_i is the genus of Σ_i. Since $n_i = O(gn)$ because of the condition for iterating the while loop and $g_i \leq g$, each iteration of the while loop takes $O(g^2(gn)\log(gn)) = O(g^3 n \log n)$ time whp, or $O(g^3 n \log^2 n)$ time in the worst case. There are at most g iterations of the while loop. □

Lemma 6 *Let α be a shortest non-separating cycle and let β be a shortest cycle crossing α exactly once. Let G' be the embedded graph obtained from G by cutting along α and attaching a disk to the boundaries. The stretch of G is the minimum between* $\text{len}(\alpha) \cdot \text{len}(\beta)$ *and the stretch of G'.*

Proof. Let Σ be the surface where G is embedded and let Σ' be the surface where G' is embedded. Since any two closed curves of G' that cross an odd number of times in Σ' also cross an odd number of times in Σ, it is clear that

$$\texttt{str}(G) \;\leq\; \min\{\texttt{str}(G'), \text{len}(\alpha) \cdot \text{len}(\beta)\}.$$

Thus, we have to argue the other inequality. If (α, β) define the stretch of G, then the other inequality is obvious.

Let us consider next the case where (α, β) do not define the stretch of G; it holds that $\texttt{str}(G) < \text{len}(\alpha) \cdot \text{len}(\beta)$. Let (γ^*, σ^*) be the pair of cycles that define the stretch of G. If there are several such pairs, we choose one such that $\texttt{cr}(\gamma^*, \alpha) + \texttt{cr}(\sigma^*, \alpha)$ is minimum. We distinguish 3 cases depending on the values of $\texttt{cr}(\gamma^*, \alpha)$ and $\texttt{cr}(\sigma^*, \alpha)$:

Case $\texttt{cr}(\gamma^*, \alpha) = \texttt{cr}(\sigma^*, \alpha) = 0$. In this case, γ^* and σ^* keep crossing once in Σ', and thus $\texttt{str}(G) = \texttt{str}(G')$.

Case $\texttt{cr}(\gamma^*, \alpha) = 1$ or $\texttt{cr}(\sigma^*, \alpha) = 1$. This case cannot actually happen. Let us assume that $\texttt{cr}(\gamma^*, \alpha) = 1$; the other case is symmetric. Since γ^* crosses α once and β is a shortest cycle crossing α once, we have $\text{len}(\beta) \leq \text{len}(\gamma^*)$. Using that α is a shortest non-separating cycle we have

$$\texttt{str}(G) \;=\; \text{len}(\gamma^*) \cdot \text{len}(\sigma^*) \;\geq\; \text{len}(\beta) \cdot \text{len}(\alpha).$$

This implies that (α, β) actually define the stretch of G.

Case $\texttt{cr}(\gamma^*, \alpha) \geq 2$ or $\texttt{cr}(\sigma^*, \alpha) \geq 2$. This case cannot actually happen. Let us assume that $\texttt{cr}(\gamma^*, \alpha) \geq 2$; the other case is symmetric. Let x and y be two crossings of γ^* and α that are consecutive along α. Because of Lemma 4, α contains a shortest path between x and y. Let π denote this shortest path. We can use π and γ^* to construct two cycles γ' and γ'' that are shorter and cross α fewer times than γ^*. Because of Lemma 1, some $\tilde{\gamma} \in \{\gamma', \gamma''\}$ crosses σ^* an odd number of times. The pair $(\tilde{\gamma}, \sigma^*)$ contradicts the choice of (γ^*, σ^*).

This finishes all cases. (Note that the second and third cases are not mutually exclusive.) □

Lemma 6 shows correctness of the algorithm if the condition $|V(G_i)| \leq g \cdot |V(G)|$ is true for each $i = 1, \ldots, g$. We next argue why we can finish the search if at some iteration $|V(G_i)| > g \cdot |V(G)|$.

Lemma 7 *If, for some i, α_i has more than $|V(G)|$ vertices, then for any $\ell \geq i$*

$$\texttt{str}(G) \;=\; \min\{\text{len}(\alpha_j) \cdot \text{len}(\beta_j) \mid j = 1, \ldots, \ell - 1\}.$$

Proof. Assume, for the sake of this proof, that in the algorithm ComputeStretch we drop testing the condition $|V(G_i)| \leq g \cdot |V(G)|$. The algorithm then makes exactly g iterations and computes cycles α_j, β_j for each $j = 1, \ldots, g$. Because of Lemma 6 it holds

$$\texttt{str}(G) \;=\; \min\{\text{len}(\alpha_j) \cdot \text{len}(\beta_j) \mid j = 1, \ldots, g\}.$$

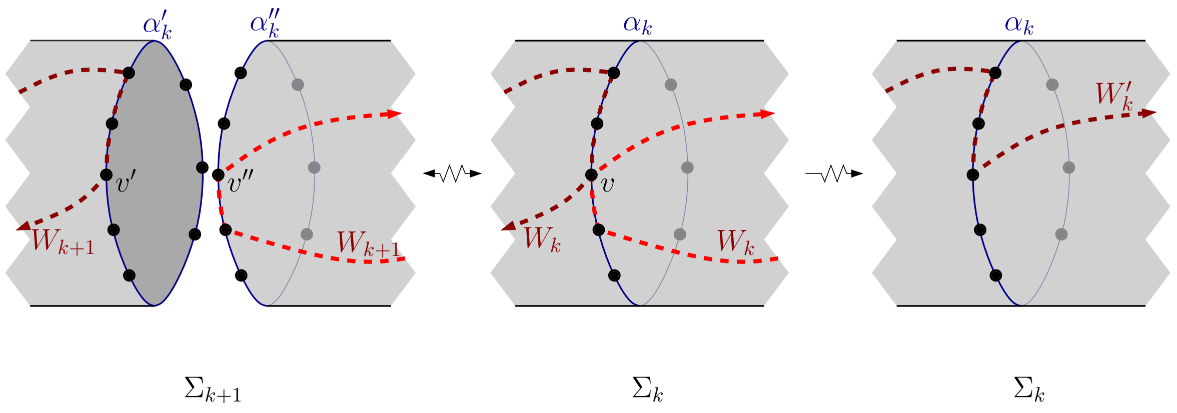

Figure 2: Figure for the proof of Lemma 7.

Assume that, at some iteration, the shortest non-separating cycle α_i in G_i, has more than $|V(G)|$ vertices. For each $k < i$, the cycle α_i corresponds to a closed walk W_k in the graph G_k. Moreover, the walk W_k does not cross the cycle α_k, for each $k < i$. Since α_i has more than $|V(G)|$ vertices, W_1 repeats some vertex of $G_1 = G$. This means that W_1 is not a cycle.

Let k be the maximum index, $1 \leq k < i$ such that W_k is *not a cycle* in G_k; thus W_{k+1} is a cycle in G_{k+1}. Since W_1 is not a cycle and W_i is a cycle, the index k is well defined. See Figure 2, left and center. Let v be a vertex of G_k that is repeated in W_k. Cutting G_k through α_k produces two copies α'_k and α''_k of α_k. Let v' and v'' be the corresponding copies of v. We can form a closed walk W'_k in G_k by taking the subwalk of W_k from the first appearance of v until the second. This closed walk W'_k crosses α_k once. Therefore $\text{len}(\beta_k) \leq \text{len}(W'_k) < \text{len}(W_k) = \text{len}(\alpha_i) \leq \text{len}(\alpha_j)$ for each $j \geq i$. We conclude that, for each $j \geq i$,

$$\text{len}(\alpha_j)\cdot\text{len}(\beta_j) > \text{len}(\beta_k)\cdot\text{len}(\alpha_j) \geq \text{len}(\beta_k)\cdot\text{len}(\alpha_k).$$

Since $k < i \leq \ell$ we conclude that

$$\begin{aligned}\texttt{str}(G) &= \min\{\text{len}(\alpha_j)\cdot\text{len}(\beta_j) \mid j = 1,\dots,g\}\\ &= \min\{\text{len}(\alpha_j)\cdot\text{len}(\beta_j) \mid j = 1,\dots,\ell-1\}.\end{aligned}$$

□

Theorem 8 *Let G be a graph with n vertices embedded in surface of genus g. We can compute the stretch of G in time $O(g^4 n \log n)$ with high probability, or in time $O(g^4 n \log^2 n)$ in the worst case.*

Proof. The time bound follows from Lemma 5. For the correctness, note that the while loop has the following invariant: at the start of iteration i, the variable *str* stores the value

$$\min(\{\text{len}(\alpha_j)\cdot\text{len}(\beta_j) \mid j = 1,\dots,i-1\} \cup \{\infty\}).$$

If $|V(G_i)| \leq g\cdot|V(G)|$ for each iteration, then the algorithm finishes with $i = g+1$, the surface Σ_{g+1} is a sphere, and correctness follows from Lemma 6. If at some iteration ℓ we have $|V(G_\ell)| > g\cdot|V(G)|$, then there was some index $i < \ell$ such that the cycle α_i had more than $|V(G)|$ vertices. Correctness then follows from Lemma 7. □

Acknowledgments. We are grateful to Daniel Štefankovič for some early discussions.

References

[1] K. J. Böröczky, J. Pach, and G. Tóth. Planar crossing numbers of graphs embeddable in another surface. *Int. J. Found. Comput. Sci.*, 17(5):1005–1016, 2006.

[2] S. Cabello, E. W. Chambers, and J. Erickson. Multiple-source shortest paths in embedded graphs, 2013. Accepted to SIAM J. Computing. Preprint available at `http://arxiv.org/abs/1202.0314`. Preliminary version at SODA 2007.

[3] S. Cabello and B. Mohar. Adding one edge to planar graphs makes crossing number hard. In *Proc. SoCG 2010*, pages 68–76, 2010. See `http://arxiv.org/abs/1203.5944` for the full version.

[4] H. Djidjev and I. Vrt'o. Planar crossing numbers of graphs of bounded genus. *Discrete & Computational Geometry*, 48(2):393–415, 2012.

[5] P. Hliněný and M. Chimani. Approximating the crossing number of graphs embeddable in any orientable surface. In *Proc. SODA 2010*, pages 918–927, 2010.