Topological Morphing of Planar Graphs*

Patrizio Angelini, Pier Francesco Cortese,
Giuseppe Di Battista, and Maurizio Patrignani

Università Roma Tre
{angelini,cortese,gdb,patrigna}@dia.uniroma3.it

Abstract. In this paper we study how two planar embeddings of the same biconnected graph can be morphed one into the other while minimizing the number of elementary changes.

1 Introduction

A useful feature of a graph drawing editor is the possibility of selecting a certain face of the drawing and of promoting it to be the external face (see, e.g., [9]). In order to preserve the mental map, the user would like that the editor executed such an operation by performing a few changes to the drawing.

The above operation is just an example of a topological feature that would be useful to have at disposal from an editor. More generally, it would be interesting to have an editor allowing the user to look at a drawing and to specify in some way, e.g. pointing at vertices or edges, a new embedding. Such an embedding could be even requested at a more abstract level, asking the editor to go to one with minimum depth, or with minimum radius, etc. Again, the editor should transform the current embedding into the new one smoothly, i.e. with the minimum number of changes.

A similar problem occurs when, keeping the topology unchanged, an editor has to geometrically morph a drawing into another one, specified in some way from the user. In this case the operations that the editor can perform are topology-preserving translations and scaling of objects. The user would like to see a geometric morphing with the minimum number of intermediate snapshots.

The existence of a geometric morphing between two drawings was addressed surprisingly long ago. Cairns proved in 1944 that between any two straight-line drawings of a triangulated planar graph there exists a morph in which any intermediate drawing is straight-line planar [7]. This was extended to general planar graphs by Thomassen in 1983 [19]. The first algorithms to find such morphings were proposed by Floater and Gotsman for triangulations [12] and by Gotsman and Surazhsky for general plane graphs [13]. While the search for a geometric morph between two given drawings of a planar graph with a polynomial number of steps and with a bounded size of the needed grid is still open, some recent studies address the problem for the special cases of orthogonal drawings [15,6] and arbitrary plane drawings [11].

* Work partially supported by MUR under Project "MAINSTREAM: Algoritmi per strutture informative di grandi dimensioni e data streams."

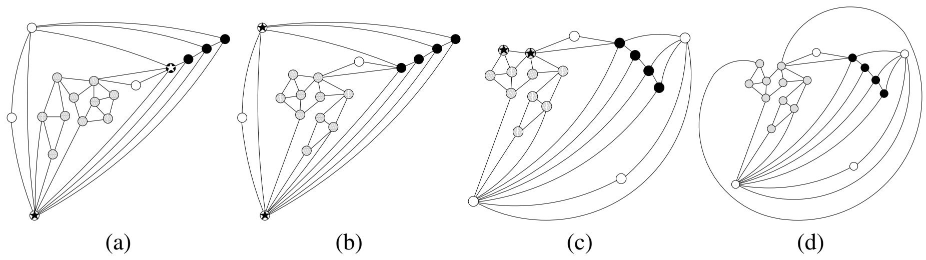


Fig. 1. A sequence of flips and skips transforming an embedding

We study the morphing between two drawings from the topological perspective and we call it *topological morphing*. There are many ways to state the problem, ranging from the family of graphs, to the operations that an editor can perform, their completeness, their ability to capture changes that are "natural" for the user, and to the metrics that distinguish a good from a bad morphing. This work starts from the following basic hypotheses. (i) We consider biconnected planar graphs, since such graphs are the building block of several graph drawing methodologies. (ii) We consider operations that move in one step entire blocks of the drawing, that are identified by some connectivity features. Namely, using a term that is common in planarity testing literature, we call *flip* the operation that allows to "flip" a component around its separation pair. Also, borrowing the term from the common rope skipping game played by children, we call *skip* the operation that allows to move the external face by "skipping" an entire component without modifying the combinatorial embedding. (iii) The metric is the number of performed operations. Namely, we have that a topological morphing is "good" if the editor performs it with a few flips and skips. Intuitively, the fewer operations are performed, the better the user preserves the mental map.

As an example, suppose that the graph is embedded as shown in Fig. 1.a and that the user would like to obtain the embedding in Fig. 1.d. A minimum sequence of operations that leads to Fig. 1.d consists of flipping the component separated by the starred vertices of Fig. 1.a, then by skipping the component separated by the starred vertices of Fig. 1.b, and finally by skipping the edge separated by the starred vertices of Fig. 1.c.

We present the following results. Let G be a biconnected planar graph and denote by $\langle \Gamma, f \rangle$ one of its combinatorial embeddings Γ with f as external face. Suppose that pair $\langle \Gamma_1, f_1 \rangle$ is the current topology and that $\langle \Gamma_2, f_2 \rangle$ represents a target topology chosen by the user. (1) In Sect. 2 we show that if both flips and skips are allowed the general problem of morphing $\langle \Gamma_1, f_1 \rangle$ into $\langle \Gamma_2, f_2 \rangle$ with the minimum number of flips and skips is NP-complete. Motivated by such a result we tackle several more restricted problems. (2) Suppose that $\Gamma_1 = \Gamma_2$ and that only skips are allowed. In Sect. 3 we give a linear time algorithm to move the external face from f_1 to f_2 with the minimum number of skips. (3) In Sect. 4 we show that the topological morphing problem can be efficiently solved if G does not have parallel triconnected components. (4) In Sect. 5 we show that the problem is fixed-parameter tractable. Basic definitions are in Sect. 2 while concluding remarks are in Sect. 6.

2 Basic Concepts

In this section we define the flip and skip operations and their properties. The proofs of the lemmas and theorems can be found in [1].

We assume familiarity with planarity and connectivity of graphs. A *planar drawing* of a graph is a mapping of its vertices to distinct points of the plane and of its edges to non-intersecting open Jordan curves between their end-points. A graph is *planar* if it has a planar drawing. A planar drawing partitions the plane into *faces* (topologically connected regions). The unbounded face is the *external face*. Two planar drawings of a graph G are *equivalent* if they determine the same circular ordering of the edges around each vertex. An equivalence class of planar drawings is a *combinatorial embedding* of G. A *planar embedding* is a pair $\langle \Gamma, f \rangle$, where Γ is a combinatorial embedding and f is the external face.

The *SPQR-tree* $\mathcal{T}$ of a biconnected graph G describes the arrangement of its triconnected components. We assume familiarity with SPQR-trees. For details see [10].

Let G be a biconnected planar graph and let $\mathcal{T}$ be the SPQR-tree of G. A planar embedding $\langle \Gamma, f \rangle$ of G can be represented by a labeling of $\mathcal{T}$. Namely, Γ is described by the combinatorial embedding of the skeleton of each node in $\mathcal{T}$, which can be succinctly represented by labeling each R-node with a Boolean value and each P-node with a circular ordering of its adjacent nodes, as described in [5].

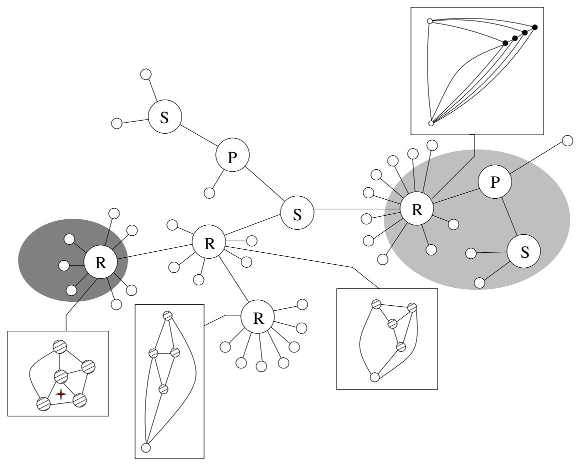
In order to account for the external face f in the SPQR-tree $\mathcal{T}$, we introduce the following definitions. A node μ of $\mathcal{T}$ is an *allocation node* of a face f of Γ either if μ is a Q-node incident to f or if there exist no virtual edge e of $skel(\mu)$ such that $pertinent(e)$ contains all the edges of f. Observe that if μ is an allocation node of f, then there is exactly one face f_μ in $skel(\mu)$ such that all the pertinent graphs of its virtual edges contain at least one edge of f. Face f_μ is the *representative* of f in $skel(\mu)$. In the following, we will denote by f both a face of Γ and its representative face in the skeleton of one of its allocation nodes. We say that f *belongs* to all its allocation nodes. The set of all the allocation nodes of f are a subtree of $\mathcal{T}$, called the *allocation tree* of f. Figure 2.a shows examples of allocation trees.

Property 1. The allocation tree of a face f is the subtree of $\mathcal{T}$ whose leaves are the Q-nodes corresponding to the edges of f.

The external face of Γ can be provided by specifying its allocation tree in $\mathcal{T}$The following lemma shows how adjacent nodes in $\mathcal{T}$ share exactly two faces.

Lemma 1. *Let μ_1 and μ_2 be two adjacent nodes of an SPQR-tree $\mathcal{T}$. There are exactly two faces f' and f'' of Γ that belong both to μ_1 and to μ_2. In $skel(\mu_1)$ ($skel(\mu_2)$) f' and f'' share edge $e(\mu_2)$ ($e(\mu_1)$). If μ_1 (μ_2) is not an S-node, then $e(\mu_2)$ ($e(\mu_1)$) is the only edge shared by f' and f'' in $skel(\mu_1)$ ($skel(\mu_2)$).*

Now we define the flip and skip operations and we show how they change the embedding of a planar graph. Let G be a planar graph, and let $\langle \Gamma, f \rangle$ be one of its embeddings. Let (u, v) be a split pair of G and let G_1 be a set of topologically contiguous maximal split components of G w.r.t. (u, v) such that G_1 does not contain all the edges of f. We define the *flip* operation on $\langle \Gamma, f \rangle$ with respect to G_1: $flip(\langle \Gamma, f \rangle, G_1) = \langle \Gamma', f' \rangle$



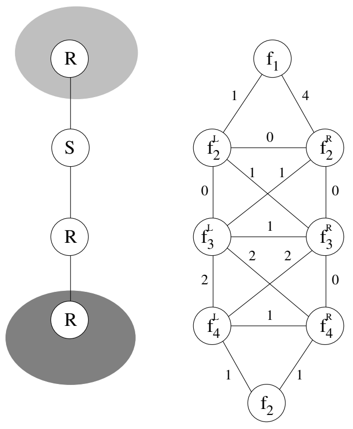


Fig. 2. SPQR-tree of the graph of Fig. 1. (a) The light gray ellipse circles the allocation tree of the external face of Fig. 1.a. The dark Gray ellipse circles the allocation tree of the external face of Fig. 1.d. (b) The skip path and the corresponding track graph of the two faces.

where Γ' is obtained from Γ by reversing the adjacency lists of all the vertices of G_1, but for u and v, and by reversing the order of the edges of G_1 in the adjacency lists of u and v. Face f' is determined as follows. If at least one of u and v is not in f, then $f' = f$. Otherwise, f' is the unique face of Γ' containing both the edges belonging to f and not belonging to G_1 and some edge of G_1 not belonging to f. As an example, see the flip applied to the embedding of Fig. 1.a that yields the embedding of Fig. 1.b.

We add the constraint that a flip operation cannot be performed if G_1 contains f because a flipping of the entire external structure of the graph around an internal component is undesirable from a comprehension point of view.

The following property describes three basic features of the flip operation and is trivial to prove.

Property 2. (a) $flip(flip(\langle\Gamma, f\rangle, G_1), G_1) = \langle\Gamma, f\rangle$. (b) If G_1 is a path, then $flip(\langle\Gamma, f\rangle, G_1) = \langle\Gamma, f\rangle$. (c) If $G - G_1$ is a path, then $flip(\langle\Gamma, f\rangle, G_1) = \langle\overline{\Gamma}, f\rangle$, where $\overline{\Gamma}$ is Γ with reversed adjacency lists.

Let $\langle\Gamma_1, f_1\rangle$ be a planar embedding of G and Γ_2 be a "target" combinatorial embedding of G. It is easy to see that there always exists a sequence of flip operations that leads from $\langle\Gamma_1, f_1\rangle$ to $\langle\Gamma_2, f_2\rangle$ for some choice of f_2 in Γ_2. We denote by $\mathcal{F}(\langle\Gamma_1, f_1\rangle, \langle\Gamma_2, f_2\rangle)$ the minimum number of flips to obtain $\langle\Gamma_2, f_2\rangle$ from $\langle\Gamma_1, f_2\rangle$ for any f_2.

Now we define the skip operation, which provides the ability to modify the external face of an embedding. Let G be a planar graph, and let $\langle\Gamma, f_1\rangle$ be one of its planar embeddings. Let (u, v) be a split pair of G incident to faces f_1 and f_2 in Γ. Skip is defined as follows: $skip(\langle\Gamma, f_1\rangle, f_2) = \langle\Gamma, f_2\rangle$. It is easy to see that there exists a sequence of skip operations that leads from $\langle\Gamma, f_1\rangle$ to $\langle\Gamma, f_2\rangle$ for any choice of f_2

in Γ. As an example, see the skip applied to the embedding of Fig. 1.b that yields the embedding of Fig. 1.c. We denote by $\mathcal{S}(\langle \Gamma, f_1\rangle, \langle \Gamma, f_2\rangle)$ the minimum number of skips to obtain $\langle \Gamma, f_2\rangle$ from $\langle \Gamma, f_1\rangle$.

Given two planar embeddings $\langle \Gamma_1, f_1\rangle$ and $\langle \Gamma_2, f_2\rangle$ of a graph G, one could ask which is the minimum number of flip and skip operations for obtaining $\langle \Gamma_2, f_2\rangle$ from $\langle \Gamma_1, f_1\rangle$. We denote by $\mathcal{FS}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle)$ such a number.

Property 3. $\mathcal{FS}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle) \leq \mathcal{F}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_3\rangle) + \mathcal{S}(\langle \Gamma_2, f_3\rangle, \langle \Gamma_2, f_2\rangle)$.

Lemma 2. *The values of $\mathcal{F}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle)$, $\mathcal{S}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_1, f_2\rangle)$, and $\mathcal{FS}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle)$ are $O(n)$, where n is the number of vertices of G.*

Unfortunately, given a biconnected planar graph G and two of its planar embeddings $\langle \Gamma_1, f_1\rangle$ and $\langle \Gamma_2, f_2\rangle$, the problem of transforming $\langle \Gamma_1, f_1\rangle$ into $\langle \Gamma_2, f_2\rangle$ with the minimum number of flip/skip operations is NP-complete.

Theorem 1. *Let G be a biconnected planar graph and let $\langle \Gamma_1, f_1\rangle$ and $\langle \Gamma_2, f_2\rangle$ be two planar embeddings of G. Both computing $\mathcal{FS}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle)$ and computing $\mathcal{F}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle)$ is NP-complete.*

3 Linearity of the Case with Fixed Combinatorial Embedding

Let G be a biconnected planar graph, and let $\langle \Gamma, f_1\rangle$ and $\langle \Gamma, f_2\rangle$ be two planar embeddings of G. In this section, we show how to compute the value of $\mathcal{S}(\langle \Gamma, f_1\rangle, \langle \Gamma, f_2\rangle)$.

First, we need to introduce the following lemma whose proof is given in [1].

Lemma 3. *Let G be a biconnected planar graph and let $\mathcal{T}$ be the SPQR-tree of G. Let $\langle \Gamma, f_1\rangle$ and $\langle \Gamma, f_2\rangle$ be two planar embeddings of G. If there exists an R-node μ of $\mathcal{T}$ such that $skel(\mu)$ contains both f_1 and f_2, then $\mathcal{S}(\langle \Gamma, f_1\rangle, \langle \Gamma, f_2\rangle)$ is the length of the shortest path from f_1 to f_2 on the dual of $skel(\mu)$.*

Let $\mathcal{T}$ be the SPQR-tree of G and let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the allocation trees of f_1 and f_2, respectively. The value of $\mathcal{S} = \mathcal{S}(\langle \Gamma, f_1\rangle, \langle \Gamma, f_2\rangle)$ can be easily computed when $\mathcal{T}_1 \cap \mathcal{T}_2 \neq \emptyset$. If this is true we have to tackle three cases: $\mathcal{T}_1 \cap \mathcal{T}_2 = \{\mu\}$, $\mathcal{T}_1 \cap \mathcal{T}_2 = \{\mu, \nu\}$, and $\mathcal{T}_1 \cap \mathcal{T}_2 = \{\mu_1, \mu_2, \dots, \mu_k\}$. Conversely, the case $\mathcal{T}_1 \cap \mathcal{T}_2 = \emptyset$ is more complex.

Case $\mathcal{T}_1 \cap \mathcal{T}_2 = \{\mu\}$. In this case μ is the only node of $\mathcal{T}$ whose skeleton contains both f_1 and f_2. If μ is an S-node, then G is a cycle and so $\mathcal{S} = 1$. If μ is a P-node, since a skip operation can move the external face from f_1 to any face of $skel(\mu)$, we have that $\mathcal{S} = 1$. Finally, if μ is an R-node, by Lemma 3, $\mathcal{S}$ is the length of the shortest path on the dual of $skel(\mu)$ from f_1 to f_2.

Case $\mathcal{T}_1 \cap \mathcal{T}_2 = \{\mu, \nu\}$. Observe that, in this case, μ and ν are adjacent in $\mathcal{T}$, and hence they cannot be both P-nodes or both S-nodes. Also, by Lemma 1, f_1 and f_2 are adjacent in both $skel(\mu)$ and $skel(\nu)$. Hence, we have that $\mathcal{S} = 1$. Notice that, if one of the two nodes, say μ, is an S-node, then all edges in $skel(\mu)$ are real edges, but for $e(\nu)$.

Case $\mathcal{T}_1 \cap \mathcal{T}_2 = \{\mu_1, \mu_2, \dots, \mu_k\}$, with $k \geq 3$. As this case is more involved, we treat it separately in the following lemma, the proof of which is left out of this extended abstract.

Lemma 4. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the allocation tree of two faces f_1 and f_2 of a graph G. If $\mathcal{T}_1 \cap \mathcal{T}_2 = \mathcal{T}_3$, $\mathcal{T}_3 = \{\mu_1, \mu_2, \ldots, \mu_k\}$, with $k \geq 3$, then $\mathcal{T}_3$ is a star graph whose central node is an S-node.*

By Lemma 4 and since f_1 and f_2 belong to the same S-node, it follows that $\mathcal{S} = 1$.

If $\mathcal{T}_1 \cap \mathcal{T}_2 = \emptyset$, the computation of $\mathcal{S}$ is not trivial; however, we provide a linear time algorithm, called SKIPONLY, to solve this problem. The algorithm is described below. We define *skip path* $sp(f_1, f_2)$ in $\mathcal{T}$ the (unique) shortest path in $\mathcal{T}$ between a node of $\mathcal{T}_1$ and a node of $\mathcal{T}_2$ (see Fig. 2.b). Since a skip operation can only move the external face from $skel(\mu)$ to $skel(\nu)$, with μ adjacent to ν, the following Property holds.

Property 4. Any sequence of skip operations that moves the external face from f_1 to f_2 must traverse all the nodes of the skip path between $\mathcal{T}_1$ and $\mathcal{T}_2$.

In order to compute the sequence of skip operations to move the external face from f_1 to f_2 with $\mathcal{S}$ steps, we define a *weighted track graph* [3] $Track(f_1, f_2)$ (see Fig. 2.b). The nodes of $Track(f_1, f_2)$ are faces of the skeletons of the nodes in $sp(f_1, f_2)$. In particular, let $\{\mu_1, \ldots \mu_k\}$ be the nodes in $sp(f_1, f_2)$, where f_1 is the external face of $skel(\mu_1)$, while f_2 is a face of $skel(\mu_k)$. Faces f_1 and f_2 are nodes of $Track(f_1, f_2)$. For each node μ_i, $i = 2, \ldots, k$, $Track(f_1, f_2)$ contains two nodes, called ${f_i}^l$ and ${f_i}^r$, corresponding to the two faces of $skel(\mu_i)$ adjacent to the virtual edge representing μ_{i-1} in $skel(\mu_i)$. Notice that such faces also correspond to the two faces of $skel(\mu_{i-1})$ adjacent to the virtual edge representing μ_i in $skel(\mu_{i-1})$. Node f_1 belongs to level 1, nodes ${f_i}^l$ and ${f_i}^r$, for $i = 2, \ldots, k$, belong to level i, and node f_2 belongs to level $k+1$.

We insert in $Track(f_1, f_2)$ two types of edges, called *horizontal edges*, connecting nodes of the same level, and *vertical edges*, connecting nodes of adjacent levels. More precisely, horizontal edges are $({f_i}^r, {f_i}^l)$, for $i = 2, \ldots, k$, with weight 1, while vertical edges are, for $i = 2, \ldots, k-1$, $({f_i}^l, {f_{i+1}}^l)$, $({f_i}^l, {f_{i+1}}^r)$, $({f_i}^r, {f_{i+1}}^l)$, $({f_i}^r, {f_{i+1}}^r)$, and edges $(f_1, {f_2}^l)$, $(f_1, {f_2}^r)$, $({f_k}^l, f_2)$, $({f_k}^r, f_2)$.

Consider a vertical edge $(f_i^{s_i}, f_{i+1}^{s_{i+1}})$, with $s_i, s_{i+1} \in \{l, r\}$, spanning levels i and $i+1$. If μ_i is a P-node, then the weight is either 0 or 1, depending on the fact that virtual edges corresponding to μ_i and μ_{i+1} are consecutive or not in the circular ordering of the nodes. If μ_i is an S-node, then the weight is either 0 or 1, depending on whether $s_1 = s_2$ or not. Finally, if μ_i is an R-node the weight is the length of the shortest path on the dual of $skel(\mu_i)$ from f_i^* to f_{i+1}^*.

The weight of an edge (f', f'') in $Track(f_1, f_2)$ represents the number of skip operations needed to move the external face from f' to f''. Weight 1 assigned to an horizontal edge $({f_i}^r, {f_i}^l)$ represents the possibility to skip the virtual edge representing μ_i in $skel(\mu_{i-1})$.

Theorem 2. *Let G be a biconnected planar graph, and let $\langle \Gamma, f_1 \rangle$ and $\langle \Gamma, f_2 \rangle$ be two planar embeddings of G. If only skip operations are allowed, then there exists an algorithm to compute $\mathcal{S}(\langle \Gamma, f_1 \rangle, \langle \Gamma, f_2 \rangle)$ in linear time.*

Proof sketch. Consider the shortest path $sp(f_1, f_2)$ on $Track(f_1, f_2)$ from f_1 to f_2 computed by Algorithm SKIPONLY. The proof is based on the fact that any sequence of skip operations leading from f_1 to f_2, by Property 4, must traverse all the levels of

$Track(f_1, f_2)$, and hence can not be shorter than the sequence identified by $sp(f_1, f_2)$. Regarding the computational complexity, since the needed operations on the SPQR-tree $\mathcal{T}$ and the sizes of involved structures are linear, it is possible to show that Algorithm SKIPONLY can be implemented to run in linear time [1]. □

4 Linearity of the Case without P-Nodes

In this section we show that if $\mathcal{T}$ does not contain P-nodes, the problem of computing $\mathcal{FS}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle)$ can be solved in linear time. For simplicity, the algorithm described in this section only considers a subset of the possible flip operations. Namely, given an S-node μ, although a legitimate flip operation may concern the split components of any split pair of μ, we only consider flip operations that concern split components of maximal split pairs of μ. Intuitively, this corresponds to flipping a single neighbor ν of μ or all the neighbors of μ with the exception of ν. At the end of the section we handle the general case.

In order to compute $\mathcal{FS}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle)$ when $\mathcal{T}$ does not contain P-nodes, we first assign a label in $\{\texttt{turned}, \texttt{unturned}\}$ to each node μ of $\mathcal{T}$. Intuitively, the label of node μ indicates whether some transformation is needed on the skeleton of μ in order to obtain Γ_2 from Γ_1. If μ is a Q-node, then μ is labeled `unturned`. If μ is an R-node, μ is labeled `unturned` if it has the same Boolean value in both the labellings representing Γ_1 and Γ_2, and `turned` otherwise. Finally, if μ is an S-node, it is labeled `unturned` (`turned`) if the majority of its adjacent R-nodes is `unturned` (`turned`). In case of a tie, we give μ an arbitrary label, unless μ is an internal S-node of the skip path sp. In this case, we give μ a label that is different from one of its adjacent R-nodes in sp.

Second, we suitably extend the labeling from the nodes to the edges. An edge e incident to a Q-node is labeled `unturned`. Otherwise, e is labeled `unturned` (`turned`) if its incident nodes have the same label (a different label). The number of `turned` edges of $\mathcal{T}$ corresponds to the minimum number of flips to be performed on Γ_1 in order to obtain Γ_2, that is $\mathcal{F}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle)$. In particular, each `turned` edge e identifies a split pair, which, since $\mathcal{T}$ has not P-nodes, identifies in its turn two split components G_1 and G_2. Any minimum sequence of flips that transforms Γ_1 into Γ_2 contains either $flip(\langle \Gamma', f'\rangle, G_1)$ or $flip(\langle \Gamma'', f''\rangle, G_2)$, for some suitable Γ', Γ'', f', and f''.

A trivial case is when the intersection of the two allocation trees $\mathcal{T}_1$ and $\mathcal{T}_2$ of f_1 and f_2 is non-empty. In such a case, since f_1 and f_2 belong to the same skeleton, there is no flip that can help to reduce the number of skips, and the trivial algorithm that first performs all flips and then all skips uses $\mathcal{FS}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle)$ operations. Since, in general, a flip operation may modify the distance between two faces (and hence modify the number of needed skips), in order to compute $\mathcal{FS}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle)$ we have to consider the case in which flip and skip operations are allowed to be alternated.

We propose an algorithm, called NOPARALLEL, to compute $\mathcal{FS}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle)$ when $\mathcal{T}_1 \cap \mathcal{T}_2 = \emptyset$ and $\mathcal{T}$ does not contain P-nodes. Such an algorithm is similar to Algorithm SKIPONLY. The weights of the edges of graph $Track(f_1, f_2)$ are modified in order to take into account the possibility of performing some flip operations in advance in order to reduce the number of skip operations. Namely, consider two nodes μ_i and

μ_{i+1} of the skip path sp, which are adjacent through the `turned` edge e, and consider a skip operation on μ_{i+1}. Such a skip operation has the effect of transferring the external face from f^l_{i+1} to f^r_{i+1} or vice versa. The same effect is obtained by flipping μ_{i+1} with respect to μ_i. Therefore, we set to 0 the weight of the horizontal edge linking f^l_{i+1} to f^r_{i+1} in graph $Track(f_1, f_2)$ and call *shortcut* such an edge. Using a shortcut in the shortest path from f_1 to f_2 corresponds to performing a flip in advance and saving a skip operation.

The sequence of skip and flip operations that transform $\langle \Gamma_1, f_1 \rangle$ into $\langle \Gamma_2, f_2 \rangle$ is given by the edges of a suitably selected weighted shortest path p from f_1 to f_2 in graph $Track(f_1, f_2)$ as follows. First, perform the flip operations corresponding to the shortcuts that are traversed by p, while the external face is still f_1. Second, perform the skip operations corresponding to the non-shortcuts edges of p. Finally, perform the flip operations corresponding to all the other `turned` edges, while the external face is f_2.

Observe that graph $Track(f_1, f_2)$ may admit more than one weighted shortest path from f_1 to f_2. Suppose that the last node of the skip path sp is a `turned` (`unturned`) R-node μ. Also suppose that a weighted shortest path p_1 from f_1 to f_2 uses an even (odd) number of shortcuts. By performing the corresponding flip operations in advance, while f_1 is the external face, the embedding of node μ will be reversed an even (odd) number of times, i.e., μ will end up `turned`. Hence, in order to obtain Γ_2, according to Property 2, we would need to perform a final flip operation with respect to an edge belonging to f_2. In this case, by using an equal cost weighted shortest path p_2 from f_1 to f_2 that traverses an odd (even) number of horizontal edges whose weight is 0 we would save the last flip. Hence, we need to compute, for any intermediate node f of $Track(f_1, f_2)$, the two weighted shortest path from f_1 to f, if both exist, using an odd and even number of shortcuts. This computation can be performed in linear time and, since all other operations can be performed in linear time, the following theorem holds.

Theorem 3. *Let G be a biconnected planar graph, and let $\langle \Gamma_1, f_1 \rangle$ and $\langle \Gamma_2, f_2 \rangle$ be two planar embeddings of G. Let $\mathcal{T}$ be the SPQR-tree of G. If $\mathcal{T}$ does not contain P-nodes then $\mathcal{FS}(\langle \Gamma_1, f_1 \rangle, \langle \Gamma_2, f_2 \rangle)$ can be efficiently computed in linear time.*

Now we show how to modify Algorithm NoParallel in order to handle the general case in which a flip operation may concern the split component of any split pair of an S-node μ. Intuitively, this corresponds to allow flipping with a single operation an arbitrary number of consecutive neighbors of μ. The general idea is to modify the SPQR-tree $\mathcal{T}$ of G, relaxing the constraint that S-nodes can not be adjacent. Namely, for any maximal sequence $\sigma_i = \nu_1, \nu_2, \dots, \nu_k$ of consecutive R-nodes with the same label adjacent to μ, we add an S-node μ_i adjacent to μ and move σ_i from the adjacency list of μ to that of μ_i. The label of μ_i is the same as the one of σ_i. The label of μ is computed as for Algorithm NoParallel.

5 Fixed Parameter Tractability of the General Case

Since transforming $\langle \Gamma_1, f_1 \rangle$ into $\langle \Gamma_2, f_2 \rangle$ is NP-complete when G is an arbitrary biconnected planar graph, in this section we study the fixed parameter tractability of the problem when the structure of G is of limited complexity.

Let $\mathcal{T}$ be the SPQR-tree of a biconnected planar graph G and let $\langle \Gamma_1, f_1\rangle$ and $\langle \Gamma_2, f_2\rangle$ be two planar embeddings of G. We present an algorithm that computes $\mathcal{FS}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle)$ in $O(n^2 \times 2^{k+h})$ time, where k and h are two parameters that describe the arrangement of P-nodes in $\mathcal{T}$ and their relationships with S-nodes.

We first describe how to handle P-nodes, which are responsible for the NP-hardness of the general problem, with a fixed parameter tractability approach. Recall that the embedding of the skeleton of each P-node μ_P is described in the labeled SPQR-trees representing Γ_1 and Γ_2 by two circular sequences of virtual edges σ_1 and σ_2, respectively. As shown in [1], the problem of morphing with the minimum number of flips σ_1 into σ_2 is equivalent to the *sorting by reversal* problem (SBR), which has been proved to be NP-hard in both cases of linear and circular sequences [8,17]. In fact, sorting virtual edges is equivalent to sorting integer numbers, where a flip of l contiguous edges corresponds to a reversal of l contiguous elements of the sequence.

The fixed parameter approach is based on the fact that SBR problem can be solved in polynomial time, both in its linear and in its circular formulation, when each number has a sign and the reversal of l contiguous elements also changes their signs [14,18,16]. Indeed, when the virtual edges of a P-node correspond to components that have to be reordered and suitably "flipped", then the problem of morphing σ_1 into σ_2 can be modeled as an instance of signed SBR problem, hence admitting a polynomial time solution. For example, if all nodes adjacent to the P-node are R-nodes, then the problem of finding the minimum number of flips that sort them is polynomial. Unfortunately, some virtual edges, as for example those corresponding to paths, do not need to be flipped in a specific way. If k such virtual edges are present, we conventionally assign to them all combinations of signs, and apply 2^k times the signed SBR polynomial algorithm. In fact, there exists an assignment of signs that make it possible to find the minimum number of flips that order a mixed signed/unsigned sequence [2].

Let $\mathcal{T}$ be the SPQR-tree of G and let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the allocation trees of f_1 and f_2, respectively. We concentrate on the case when $\mathcal{T}_1 \cap \mathcal{T}_2 = \emptyset$ that is the most complex.

In order to compute $\mathcal{FS}(\langle \Gamma_1, f_1\rangle, \langle \Gamma_2, f_2\rangle)$ each node of $\mathcal{T}$ is labeled as `turned`, `unturned`, or `neutral`. We order them based on their distance from sp. First, starting from the farthest ones, we label nodes that are not in sp with the strategy described below. Second, we label nodes of sp with a different strategy. Consider the current unlabeled node μ not in sp. Observe that μ has all labeled adjacent nodes with the exception of the node that links μ to sp. If μ is an R-node, then we label μ based on its embedding as described in Algorithm NoParallel. If μ is a Q-node, we label μ `neutral`. If μ is an S-node, we assign μ the label of the majority of its non-neutral labeled adjacent nodes. In case of a tie, we label μ `neutral`. If μ is a P-node, denote by σ_1 and σ_2 the two circular sequences representing the embedding of μ in Γ_1 and Γ_2. While labeling μ, we also compute the flips that are needed to transform σ_1 into σ_2. Observe that, since the external face can not be internal to a subgraph that is flipped, σ_1 and σ_2 are actually linear sequences as far as flip operations are concerned. In particular, we denote by σ'_1 and σ'_2 the two linear sequences obtained from σ_1 and σ_2, respectively, by removing the virtual edge e corresponding to the node that links μ to sp and starting from the virtual edge following e in the sequence. Let k be the number of `neutral` elements of σ'_1 and σ'_2. We assign all possible combinations of `turned` and `unturned` values to them, and compute 2^k times the linear signed SBR distance d from σ'_1 to σ'_2, and the analogous distance $\overline{d}$ from σ'_1 to $\overline{\sigma}'_2$, where $\overline{\sigma}'_2$ is obtained from σ'_2 by reversing the order and changing the signs. If $d < \overline{d}$ ($d > \overline{d}$, $d = \overline{d}$, respectively) we assign μ the label `unturned` (`turned`, `neutral`, respectively).

Now we describe how to assign labels to the elements of the skip path $sp = \mu_1, \mu_2, \ldots, \mu_m$ from $\mathcal{T}_1$ to $\mathcal{T}_2$. Nodes in sp are never labeled `neutral`. If we have h P-nodes in sp, we consider for them all the combinations of the two possible values `turned` and `unturned`, and we repeat 2^h times the computation that follows. R-nodes and S-nodes of sp are labeled as described in Sect. 4. Analogously to Algorithm NOPARALLEL, we extend the labeling to the edges of $\mathcal{T}$. In particular, an edge is labeled `turned` if it links a `turned` node to an `unturned` one and such nodes are not P-nodes, otherwise is labeled `unturned`. We construct a *weighted track graph* $Track(f_1, f_2)$ as in Algorithm NOPARALLEL where P-nodes were not present, and we describe how to set the weights of the edges exiting nodes f_i^l and f_i^r corresponding to a P-node μ_i of sp. All other weights are set as described in Sect. 4. Denote by σ_1 and σ_2 the two circular sequences representing the embedding of μ_i in Γ_1 and Γ_2. From σ_1 we obtain the linear sequence σ_1^l (σ_1^r) ending with (starting with, respectively) the virtual edge corresponding to μ_{i-1}. Intuitively, sequence σ_1^l (σ_1^r) corresponds to the configuration of the parallel component when the external face is f_i^l (f_i^r). Analogously, from σ_2 we obtain the linear sequence σ_2^l (σ_2^r) ending with (starting with, respectively) the virtual edge corresponding to μ_{i+1}. Our aim is to set the weight of each vertical edge (f_i^s, f_{i+1}^t), for $s, t \in \{l, r\}$, as the minimum number of operations needed to transform σ_1^s into σ_2^t. Observe that, when the external face is moved from f_i^s to another face f_i of $skel(\mu_i)$ in Γ_1, we obtain a new linear sequence σ_1^* with the same circular order as σ_1^s. Namely, σ_1^* is obtained from σ_1 by opening it between the two virtual edges adjacent to f. Hence, when computing the minimum number of operations needed to transform σ_1^s into σ_2^t, we have to consider the possibility to first transforming σ_1^s into another linear sequence σ_1^* with the same circular order, that can be done by performing one skip operation, and then transforming σ_1^* into σ_2^t with the minimum number of flips, that can be done by applying the signed SBR algorithm. In order to do this, observe that all nodes adjacent to μ_i in $\mathcal{T}$ are labeled as `turned`, `unturned`, or `neutral`. Let k be the number of nodes adjacent to μ_i and labeled `neutral`. As described above, we consider all possible assignments of `turned` and `unturned` values to such nodes, and we compute 2^k times the linear signed SBR distance from σ_1^* to σ_2^t. The weight of vertical edge (f_i^s, f_{i+1}^t) is the minimum of such $n_i \times 2^k$ values, where n_i is the number of nodes adjacent to μ_i in $\mathcal{T}$. The weight of an horizontal edge for a P-node is 1.

The remaining part of the algorithm strictly follows the lines of Algorithm NOPARALLEL. Namely, we compute the minimum weight path from f_1 to f_2 in $Track(f_1, f_2)$ and, based on such a path, we decide the sequence of skip and flip operations to be performed. Again, if $Track(f_1, f_2)$ admits more than one minimum weight path, we choose among such paths taking into account the number of shortcuts traversed, corresponding to flip operations that are convenient to be performed in advance.

Here we analyze the computational complexity of the algorithm. All the operations, except those involving P-nodes, can be performed in linear time. For each P-node μ_i not belonging to the skip path, the computation of the minimum number of flips that are needed to transform σ_1 into σ_2 can be performed in $O(n_i \times 2^k)$, where n_i is the number of neighbors of μ_i in $\mathcal{T}$. Observe that computing the minimum SBR distance can be done in linear time [4], while actually finding the sequence of operations that yield that minimum can be done in time $O(n^{\frac{3}{2}}\sqrt{log(n)})$ time [18]. Hence, when considering the 2^k assignments, we only compute the distance and then, when the optimal assignment has been found, we perform the algorithm for finding the actual sequence of flips. For each P-node μ belonging to sp, the computation of the minimum number of flips that are needed to transform σ_1^s into σ_2^t can be performed in $O(n_i^2 \times 2^k)$. Namely, we have to consider the 2^k assignments of signs to the k `neutral` neighbors of μ_i and the possibility to transform σ_1^s into σ_2^t by first moving the external face to each of the n_i faces of $skel(\mu_i)$ in Γ_1 and then performing the computation of the signed linear SBR distance in linear time. Since such a computation has to be performed for each of the 2^h assignments of labels to the h P-nodes of sp, the global computational complexity of the algorithm is $O(2^h \times \sum_{i=1}^{h}(n_i^2 \times 2^{k+h}))$, which is equal to $O(n^2 \times 2^{k+h})$, since the total number of neighbors of all the P-nodes is less or equal than the total number of edges of $\mathcal{T}$, that is $O(n)$. Based on the above discussion we have:

Theorem 4. *Let G be a biconnected planar graph, let $\langle\Gamma_1, f_1\rangle$ and $\langle\Gamma_2, f_2\rangle$ be two planar embeddings of G. Let $\mathcal{T}$ be the SPQR-tree of G, let k be the maximum number of neutral S-nodes adjacent to a P-node in $\mathcal{T}$, and let h be the number of P-nodes in the skip path $sp(f_1, f_2)$. If both flip and skip operations are allowed, then $\mathcal{FS}(\langle\Gamma_1, f_1\rangle, \langle\Gamma_2, f_2\rangle)$ can be computed in $O(n^2 \times 2^{k+h})$ time.*

6 Conclusions

Preserving the user mental map while coping with ever-changing information is a common goal of the Graph Drawing and the Information Visualization areas. The information represented, in fact, may change with respect to three different levels of abstraction: (i) structural changes may modify the graph that the user is inspecting; (ii) topological changes may affect the way the same graph is embedded on the plane; and (iii) drawing changes may map the same embedded graph to differently positioned graphic objects. A large body of literature has been devoted to structural changes, addressing the representation models and techniques in the so-called dynamic and on-line settings. Also, much research effort has been devoted to manage drawing changes, where the target is to preserve the mental map by morphing the picture while avoiding intersections and overlappings. On the contrary, to our knowledge, no attention at all has been devoted to topological changes, that is, changes of the embedding of a graph in the plane.

In this paper we addressed the topological morphing problem. Namely, the problem of morphing a topology into another one with a limited number of changes. This paper leaves many open problems. (1) Primitives. We considered two topological primitives, called flip and skip. It would be important to enrich such a set with other operations that can be considered "natural" for the user perception. (2) Connectivity. It is easy to extend the results presented in Sect. 3 to simply connected graphs. However, the other presented results are deeply related to biconnectivity. There is a lot of space here for further investigation. (3) We gave the same weight to the operations performed by the morphing. However, other metrics are possible. One could weight an operation as a non-decreasing function of the moved edges or of the thickness of the moved component.

As a final remark we underline how usually the Computational Biology field looks at Graph Drawing as a tool. In this paper it happened the opposite. In fact, Theorems 1 and 4 exploit Computational Biology results.

References

1. Angelini, P., Cortese, P.F., Di Battista, G., Patrignani, M.: Topological morphing of planar graphs. Tech. Report RT-DIA-134-2008, Dept. of Computer Sci., Univ. di Roma Tre (2008)
2. Auyeung, A., Abraham, A.: Estimating genome reversal distance by genetic algorithm. CoRR cs.AI/0405014 (2004)
3. Bachmaier, C., Brandenburg, F.J., Forster, M.: Track planarity testing and embedding. In: Van Emde Boas, P., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2004. LNCS, vol. 2932, pp. 3–17. Springer, Heidelberg (2004)
4. Bader, D.A., Moret, B.M.E., Yan, M.: A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. Journal of Computational Biology 8(5), 483–491 (2001)
5. Bertolazzi, P., Di Battista, G., Didimo, W.: Computing orthogonal drawings with the minimum number of bends. IEEE Transactions on Computers 49(8), 826–840 (2000)
6. Biedl, T.C., Lubiw, A., Spriggs, M.J.: Morphing planar graphs while preserving edge directions. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS, vol. 3843, pp. 13–24. Springer, Heidelberg (2006)
7. Cairns, S.S.: Deformations of plane rectilinear complexes. American Math. Monthly 51, 247–252 (1944)
8. Caprara, A.: Sorting by reversals is difficult. In: RECOMB 1997: Proceedings of the first annual international conference on Computational molecular biology, pp. 75–83. ACM Press, New York (1997)
9. de Fraysseix, H., Ossona de Mendez, P.: P.I.G.A.L.E - Public Implementation of a Graph Algorithm Library and Editor, sourceForge project page, `http://sourceforge.net/projects/pigale`
10. Di Battista, G., Tamassia, R.: On-line planarity testing. SIAM J. Comput. 25, 956–997 (1996)
11. Erten, C., Kobourov, S.G., Pitta, C.: Intersection-free morphing of planar graphs. In: Liotta, G. (ed.) GD 2003. LNCS, vol. 2912, pp. 320–331. Springer, Heidelberg (2004)
12. Floater, M., Gotsman, C.: How to morph tilings injectively. Journal of Computational and Applied Mathematics 101, 117–129 (1999)
13. Gotsman, C., Surazhsky, V.: Guaranteed intersection-free polygon morphing. Computers and Graphics 25, 67–75 (2001)
14. Kaplan, H., Shamir, R., Tarjan, R.E.: Faster and simpler algorithm for sorting signed permutations by reversals. In: SODA 1997, pp. 344–351 (1997)
15. Lubiw, A., Petrick, M., Spriggs, M.: Morphing orthogonal planar graph drawings. In: SODA 2006, pp. 222–230. ACM Press, New York (2006)
16. Meidanis, J., Walter, M., Dias, Z.: Reversal distance of sorting circular chromosomes. Tech. Report IC-00-23, Institute of Computing, Universidade Estadual de Campinas (2000)
17. Solomon, A., Sutcliffe, P., Lister, R.: Sorting circular permutations by reversal. In: Dehne, F., Sack, J.-R., Smid, M. (eds.) WADS 2003. LNCS, vol. 2748, pp. 319–328. Springer, Heidelberg (2003)
18. Tannier, E., Bergeron, A., Sagot, M.F.: Advances on sorting by reversals. Discrete Appl. Math. 155(6-7), 881–888 (2007)
19. Thomassen, C.: Deformations of plane graphs. Journal of Combinatorial Theory, Series B 34, 244–257 (1983)