# A Simple Linear-Time Algorithm to Find the Contour in a Coloured Triangular Graph

Carsten Dorgerloh*, Jens Lüssem †
Institut für Informatik der Universität Bonn
Römerstr. 164, D-53117 Bonn
Germany

**Abstract**

We develop an $\mathcal{O}(n)$ algorithm to construct the contour of an $n$-vertex coloured triangular graph.

## 1 Introduction

Informally, the contour problem is defined as follows: Given an embedding of a triangular planar graph $T = (V_T, E_T)$, where each face is either coloured black or white, we want to compute the set of cycles $C$ defined by edges of $T$, which are on a common boundary of a black and a white triangle. Furthermore, $C$ is constrained to contain only those cycles which are not enclosed by another cycle.
The contour of a graph finds industrial applications in topology design [CSJ 93].
The paper is organized as follows. Section 2 contains a precise definition of the problem for the 2-dimensional case and some definitions and notations used throughout the paper. Section 3 describes the elementary steps of the algorithm for the contour problem and shows how to implement them efficiently. From the proof of correctness of the elementary steps the correctness of the main algorithm of the sequence is immediately clear. Finally, we describe that the algorithm can easily be extended to the $d$-dimensional case.

## 2 Notations and definitions

The terminology used in this paper follows that of Even [Ev 79]. Let $G = (V, E)$ be a planar graph. Consider a fixed plane embedding of G. The unbounded region is called the *exterior face*. Other faces are called *interior faces*. The vertices and the edges on the exterior face are called *exterior vertices* and *exterior edges*, respectively. For each vertex $v$, $N(v)$ denotes the set of neighbours of $v$. A planar graph $T = (V_T, E_T)$ has a *triangular embedding*, if every face of $T$, except the exterior face, is a triangle. The triangles of a *coloured triangular embedding* are coloured black and white, respectively. The colour of the external face is always assumed to be white. Let $S$ be a set that contains all edges of $T$ which are on a common boundary of a black and a white triangle. In fact, $S$ is a

*email: carsten@cs.uni-bonn.de
†email: jens@cs.uni-bonn.de

```
extend(G)
   V := V ∪ {v_ext}
   for each u ∈ V − {v_ext} do
      if degree(u) < 3 then
         E := E ∪ {u, v_ext}
      end if
   end for
```

Figure 1: Extend $G$ by $v_{ext}$

collection of edge-disjoint cycles (the points of the cycles being evident by context) of $T$, which are separating white and black regions. An *(external) contour* $C$ of a coloured triangular embedding $T$ consists of those cycles of $S$, which are not enclosed by another cycle of $S$. The task of computing the external contour of $T$ can now be formalized to produce the set $C$.

The following lemma is helpful, because for planar graphs it implies that an $\mathcal{O}(|V|+|E|)$ algorithm is really an $\mathcal{O}(|V|)$ algorithm. A proof can be found in [Ev 79].

**Lemma 1** *If $G = (V, E)$ is any planar graph with $|V| \geq 3$. Then $G$ has at most $3|V| - 6$ edges.*

# 3 The building blocks

The algorithm is built of the following steps.

## 3.1 Construction of the dual graph

Given a coloured triangular embedding $T$, its *coloured dual* $G = T^*$ is constructed as follows: simply trace the boundary of each face, place a vertex in $G$ for each face of $T$ (excluding the exterior face) and assign the corresponding colour to it. If two faces of $T$ have an edge $e_T$ in common, join the corresponding vertices in $G$ by an edge $e$.

It is quite straightforward to solve the problem in $\mathcal{O}(|V|)$ time sequentially, if we trace the boundaries by following cyclic linked lists.

In the following we denote by $construct\_dual(T)$ the procedure that executes the step as described above.

## 3.2 The extension of the dual

Now, we extend $G$ by introducing a new vertex $v_{ext}$. $v_{ext}$ corresponds to the external face of $T$ and is assigned the colour white. We connect this vertex to each node $u \in V - \{v_{ext}\}$ with $degree(u) < 3$ (see Figure 1). That are exactly those vertices in $G$ corresponding to faces in $T$, which have a common boundary with the external face of $T$. Hence, adding $v_{ext}$ cannot destroy the planarity of $G$.

```
hollow_out(G)
    S := N(v_ext)
    while S ≠ ∅ do
        u := first element of S
        if colour(u) = white then
            for each v ∈ N(u) − {v_ext} do
                if v_ext ∉ N(v) then
                    E := E ∪ {v, v_ext}
                E := E − {v, u}
                S := S ∪ {v}
            end for
            E := E − {u, v_ext}
            V := V − {u}
        else
            mark(u)
        end if
        S := S − {u}
    end while
```

Figure 2: Formulation of the procedure *hollow_out*

### 3.3 The hollow out step

What is the purpose of introducing the special vertex $v_{ext}$? The reason is, that we are now able to hollow out the white "regions" of $T$ starting at the external face of $T$. This is done by changing $G$, while $T$ remains unchanged. Furthermore, black vertices corresponding to faces in $T$ which contribute edges to the contour of $T$ are marked. The procedure which executes this step is given in Figure 2.

Before we prove the correctness of $hollow\_out(G)$, we need the following definition. We say a vertex $u \in V$ is *enclosed by black*, if there is no path from $v_{ext}$ to $u$ in $G$ such that all vertices on that path, except maybe $u$, are coloured white.

**Lemma 2** *$hollow\_out(G)$ applied to the dual $G = (V, E)$ (extended by $v_{ext}$) of any triangular embedding of $T = (V_T, E_T)$ requires $\mathcal{O}(|V|)$ time.*

PROOF: $S$ initially contains all neighbours of $v_{ext}$. In the course of the algorithm the set $S$ is modified as follows. In each execution of the while-loop one vertex $u \in S$ is picked. If the colour of $u$ is white, then all neighbours of $u$, except $v_{ext}$, are inserted into $S$. Furthermore, $G$ is changed: each $v \in N(u) - \{v_{ext}\}$ is connected to $v_{ext}$, $u$ and all incident edges are deleted from $G$. On the other hand, if the colour of $u$ is black, then $u$ is marked. Finally, $u$ is deleted from $S$. It follows by an inductive argument, that every white vertex which is not enclosed by black is deleted from $G$. Moreover, every black vertex which is not enclosed by black is marked.

Lemma 1 guarantees that the while loop is executed at most $\mathcal{O}(|V|)$ times. Each of the

```
remove_v_ext(G)
    for all u ∈ N(v_ext)
        E := E − {u, v_ext}
    end for
    V := V − {v_ext}
```

Figure 3: Removal of $v_{ext}$

$\mathcal{O}(|V|)$ runs of the while loop needs constant time because each vertex $u \in V - \{v_{ext}\}$ has at most three neighbours. ■

### 3.4 Finding the black components

In this step we first remove the vertex $v_{ext}$ and all edges which are incident to that node from $G$ (see Figure 3). The remaining graph is made up of what we call the *black components* of $G$. The computation of the black components can be done using standard algorithms for connected components which are based on depth-first-search or breadth-first search (see e.g. [AHU 83]). This step runs in $\mathcal{O}(max(|V|, |E|))$ time and is denoted by $compute\_black\_components(G)$.

### 3.5 Computation of the external contour

We describe the procedure $contour\_of\_component(G_i, T, G, G_{init})$, which computes the external contour of a component $G_i$ of $G$. The structure of the procedure is best described by a listing (see Figure 4). Again, we need several definitions. Each $u \in V$ corresponds to a face in $T$. Let us denote by $\lambda(u)$ the set of vertices and by $\gamma(u)$ the set of edges of the corresponding triangle in $T$. By $G_{init}$ we denote the graph produced by $construct\_dual(T)$.

**Lemma 3** *The edge list returned by $contour\_of\_component(G_i, T, G, G_{init})$ is the external contour of $G_i$. $contour\_of\_component(G_i, T, G, G_{init})$ runs in time $\mathcal{O}(|V|)$.*

PROOF: Since only the faces corresponding to marked vertices contribute edges to the external contour $E_{contour}(i)$ of $G_i$, it suffices to investigate only such vertices. Let $u$ be a marked vertex with $degree_{G_{init}}(u) = 3$ and $v \in N_{G_{init}}(u)$. Let $e_c \in E_T$ be the common edge of the faces corresponding to the vertices $u$ and $v$, respectively. Furthermore, assume that $colour(v) = white$. We claim that $e_c \in E_{contour}(i)$. Suppose, to the contrary, that $e_c \notin E_{contour}(i)$. As an immediate consequence, the other edges of $\gamma(u)$ are not in $E_{contour}(i)$. But this implies that $u$ is not marked, which is a contradiction. The other cases consider exterior edges and can be proven similarly. Since the computation of a common edge of two faces takes $\mathcal{O}(1)$ time, $contour\_of\_component(G_i, T, G, G_{init})$ runs in time $\mathcal{O}(|V|)$. ■

```
contour_of_component(G_i, T, G, G_init)
  M := {u ∈ V_i | u is marked}
  E_contour(i) := V_contour(i) := ∅
  for each u ∈ M do
    S := V_aux := ∅
    if degree_G_init(u) = 3 then
      for each v ∈ N_G_init(u) do
        if colour(v) = white then
          for each u_T ∈ λ(u), v_T ∈ λ(v) do
            if u_T = v_T then V_contour(i) := V_contour(i) ∪ {u_T}
            if |V_contour(i)| = 2 then E_contour(i) := E_contour(i) ∪ {Pop(V_contour(i)), Pop(V_contour(i))}
          end for
        end if
      end for
    else if degree_G_init(u) = 2 then
      for each v ∈ N_G_init(u) do
        for each u_T ∈ λ(u), v_T ∈ λ(v) do
          if u_T = v_T then
            if u_T ∈ V_aux then inner_point := u_T
            V_aux := V_aux ∪ {u_T}
            if colour(v) = white then V_contour(i) := V_contour(i) ∪ {u_T}
            if |V_contour(i)| = 2 then E_contour(i) := E_contour(i) ∪ {Pop(V_contour(i)), Pop(V_contour(i))}
          end if
        end for
      end for
      V_aux := V_aux − {inner_point}
      E_contour(i) := E_contour(i) ∪ {Pop(V_aux), Pop(V_aux)}
    else if degree_G_init(u) = 1 then
      for each u_T ∈ λ(u) do
        S := S ∪ {u_T}
        for each v_T ∈ λ(u) − S do
          E_contour(i) := E_contour(i) ∪ {u_T, v_T}
        end for
      end for
      v := neighbour of u in G_init
      if colour(v) = black then
        for each u_T ∈ λ(u), v_T ∈ λ(v) do
          if u_T = v_T then V_aux := V_aux ∪ {u_T}
          if |V_aux| = 2 then E_contour(i) := E_contour(i) − {Pop(V_aux), Pop(V_aux)}
        end for
      end if
    end if
  end for
```

Figure 4: Formulation of the procedure $contour\_of\_component(G_i, T, G, G_{init})$

$external\_contour()$
  $construct\_dual(T)$
  $extend(G)$
  $hollow\_out(G)$
  $remove\_v\_ext(G)$
  $compute\_black\_components(G)$
  `for each black component` $G_i$ `of` $G$
    $contour\_of\_component(G_i, T, G, G_{init})$

Figure 5: The main procedure

# 4 The main procedure

Now we present the main procedure, *external_contour* (see Figure 5), putting the developed things together. We summarize the analysis of the previous section in the following theorem.

**Theorem 4** *The external contour of a triangular graph can be computed in time* $\mathcal{O}(|V|)$.

An algorithm with this complexity is given explicitely.

# 5 Extension to the $d$-dimensional case

The above algorithm generalizes to the $d$-dimensional ($d > 2$) case where we consider $d$-simplexes.
First, we illustrate the changes neccessary by describing the 3-dimensional case. Here, we have to consider tetrahedrons instead of triangles. The dual graph is now constructed by identifing each tetrahedron by a vertex. Two vertices are joined by an edge, if their corresponding tetrahedrons have a face in common. Thus, each vertex in the dual graph has at most four neighbours. The adaption of the other steps is straightforward and it can easily be shown that the algorithm runs in time linear in the number of tetrahedrons. This method can be extended to the $d$-dimensional ($d > 3$) case as well: Two $d$-simplexes are adjacent if they have a $(d-1)$-simplex in common. A vertex in the dual graph has at most $d+1$ neighbours. Again, it can be shown that the algorithm runs in time linear in the number of $d$-simplexes.

# 6 Acknowledgements

We thank Christoph Hundack and Marc Schoenauer for stimulating remarks and discussions.

## References


[AHU 83] Aho, A., Hopcroft, J., Ullman, J., *Data Structures and Algorithms*, Addison-Wesley Publishing Company, 1983.

[CSJ 93] Chapman, C., Saitou, K., Jakiela, M., *Genetic Algorithms as an Approach to Configuration and Topology Design*, ASME Design Automation Conference, Albuquerque (1993).

[Ev 79] Even, S., *Graph Algorithms*, Computer Science Press, 1979.