C-Planarity of Clustered Graphs: Methodologies, Algorithms, and Applications

Pier Francesco Cortese
C-Planarity of Clustered Graphs:
Methodologies, Algorithms, and Applications

A thesis presented by
Pier Francesco Cortese
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Engineering
Roma Tre University
Dept. of Informatics and Automation
March 2007
COMMITTEE:
Prof. Giuseppe Di Battista

REVIEWERS:
Prof. Rosanna Petreschi
Prof. Sue H. Whitesides
To my father and mother
Acknowledgments

There are many people I would like to thank at the end of the PhD course.

First of all, I would like to thank my advisor, Giuseppe Di Battista, who always believed in me since I was a student at Roma Tre university. Working with him in his research group, I learnt how fascinating are the research and the university life. A special thanks is also due to Titto and Pizzo, for their help and precious suggestions during the PhD.

I also would like to thank the friends of University of Perugia, for their friendship and the useful and interesting collaboration during these years.

I have to thanks my colleagues and friends of our research group: a special thanks to Max, for his friendship and his help since we start together the PhD course.

Many other people deserve to be thanked, even if they are not directly involved in the work of this thesis.

I thanks my brother Gian Paolo and all my relatives, for their steady support and encouragement in my career and in all my life.

I would like to thank our students Chiara and Giulio, because I learnt also from them the appeal of teaching.

Of course, I cannot forget to thank my "historical" friend Stefano, and all the new and old friends of Roma Tre: Giulio, Pietro, Paola, Sonia, Tommaso, Elena, Gian Paolo S, Saverio.

I also would like to thank many other friends, who shared with me several initiatives in these years: Giuseppe S, Rosanna, Elisa, Luciano, Monte, Chiara, Maria Chiara, Maria Concetta, Licia, Elena, Francesco, Francesca, Emilia.

I cannot forget several friends who live near Rome ("dei Castelli"): an hearty thanks to Paolo, Sara, Andrea, Fernando, Barbara, and of course to my best friend Federica, to her husband Giuseppe and to the young Pietro.

Finally, I special thanks to Gian Paolo G, Federica, Paola, Renato, to all friends of "school of community", and to Don Julian Carron.
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Contents</td>
<td>ix</td>
</tr>
<tr>
<td></td>
<td>List of Figures</td>
<td>xi</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Background</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Graphs</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Connectivity</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>Drawings and Planarity</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>SPQR-trees</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>Clustered Graphs</td>
<td>15</td>
</tr>
<tr>
<td>2.6</td>
<td>State of the Art of C-Planarity</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>A Characterization of C-Planarity for C-Connected Graphs</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>Characterization in the Biconnected Case</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Characterization of the C-Planarity of General C-connected Clustered</td>
<td>34</td>
</tr>
<tr>
<td>4</td>
<td>A Linear Time Test for C-Connected Graphs</td>
<td>39</td>
</tr>
<tr>
<td>4.1</td>
<td>Testing and Embedding Algorithm: Biconnected Case</td>
<td>39</td>
</tr>
<tr>
<td>4.2</td>
<td>Testing and Embedding Algorithm: General Case</td>
<td>47</td>
</tr>
<tr>
<td>5</td>
<td>Cycles of Clusters</td>
<td>59</td>
</tr>
<tr>
<td>5.1</td>
<td>Preliminaries</td>
<td>60</td>
</tr>
<tr>
<td>5.2</td>
<td>Cycles with Three Clusters</td>
<td>61</td>
</tr>
<tr>
<td>5.3</td>
<td>Clusters and Grammars</td>
<td>71</td>
</tr>
<tr>
<td>5.4</td>
<td>Cycles in Cycles of Clusters</td>
<td>73</td>
</tr>
</tbody>
</table>
## 6 Clustered Cycles

6.1 The Problem ........................................... 79
6.2 Basic Definitions ....................................... 81
6.3 Fountain Clusters ....................................... 82
6.4 C-Planarity Testing of Clustered Cycles ........ 89
6.5 Computing C-Planar Embeddings of Clustered Cycles .... 95

## 7 An Example Application

7.1 The Visualization Problem ............................ 97
7.2 Background ............................................. 99
7.3 Choosing the Visualization Metaphor .............. 104
7.4 Layout Algorithm .................................... 106
7.5 Experimental Evaluation .............................. 112

Conclusions .............................................. 121

Bibliography ............................................. 125
List of Figures

2.1 (a) A graph $G$ with 7 vertices. (b) The subgraph of $G$ induced by vertices $v_1, v_2, v_3, v_6$ and $v_7$. .......................................................... 6

2.2 (a) A connected graph that is not biconnected; $u$ and $v$ are two distinct cutvertices. (b) A biconnected graph that is not triconnected; $\{v, w\}$ and $\{v, z\}$ are two distinct separation pairs. (c) A triconnected graph. ............... 7

2.3 (a) A tree. (b) A rooted tree with depth 3; node $\rho$ is the root of the tree; node $\mu$ is a child of $\beta$; the leaves are the white nodes. ......................... 8

2.4 (a) A graph and its blocks; each block is bounded by a dashed region; the numbered light grey vertices are cutvertices. (b) The block cutvertex tree of the graph in (a); the B-nodes are the big circles. ......................... 9

2.5 A planar drawing. ................................................................. 11

2.6 Three planar drawings of the same graph. The drawings in (a) and (b) are equivalent. ................................................................. 12

2.7 An embedded graph and its dual. The dual graph has light grey vertices and edges. ................................................................. 13

2.8 A planar graph (a) and the corresponding SPQR-tree (b). The black nodes of the tree represent the Q-nodes, while the dashed lines in the skeletons represent the reference edges. ................................. 20

2.9 An example of a c-connected clustered graph $C$. (a) A c-planar drawing of $C$. (b) The inclusion tree of $C$. ................................. 21

2.10 An example of a clustered graph $C$ which is non-c-connected. (a) A c-planar drawing of $C$. (b) The inclusion tree of $C$. ......................... 21

2.11 An example of a non c-planar clustered graph. ............................. 22

2.12 The classes of clustered graphs for which the c-planarity testing problem can be solved in polynomial time. The green areas indicate the classes of clustered graphs described in this thesis. ................................. 22
2.13 An example of a flat clustered graph C containing 6 clusters (plus the root). (a) A c-planar drawing of C. (b) The inclusion tree of C. (c) The graph of the clusters of C. ................................. 23

3.1 Clustered graph. (a) Underlying graph. (b) Inclusion tree. (c) SPQR-tree. The boxes contain the skeletons of selected nodes. The triple on each virtual edge represents \( lcc(e) \), \( lsc(e) \), and \( hsc(e) \), respectively. Faces of skeletons are labeled with their \( lcc \).............................................. 36

3.2 Lowest connecting paths and clusters. (a) A virtual edge \( e = (u,v) \). (b) Graph \( pertinent(e) \); the thick lines show the lowest connecting path \( p_1 \). (c) Tree \( T \) restricted to the clusters in \( pertinent(e) \)...................... 37

3.3 Illustration for the proof of Lemma 2. The paths \( s_1 \) and \( s_2 \) are drawn in thick lines, while the subpaths \( \tilde{s}_1 \) and \( \tilde{s}_2 \) are drawn in dashed lines. .......... 37

3.4 (a) A virtual edge \( e = (u,v) \). (b) Graph \( pertinent(e) \). (c) Tree \( T \) restricted to the clusters in \( pertinent(e) \). We have that \( lcc(e) = \delta \), \( lsc(e) = \beta \), and \( hsc(e) = \alpha \). If the edge \( e' \) is removed from \( pertinent(e) \) then \( lsc(e) \) becomes \( \delta \). ................................................................. 38

4.1 (a) A portion of the BC-tree for the proof of Lemma 15. (b) The relationships between three subgraphs \( pertinent(\mu) \), \( pertinent(\alpha, u) \), and \( pertinent(\alpha) \), denoted \( p(\mu) \), \( p(u, \alpha) \) and \( p(\alpha) \), respectively. ......................... 48

4.2 The c-planarity testing and embedding algorithm for c-connected clustered graphs whose underlying graph is biconnected - Part 1. ................... 52

4.3 The c-planarity testing and embedding algorithm for c-connected clustered graphs whose underlying graph is biconnected - Part 2. ............ 53

4.4 Testing and embedding procedure for \( P \) nodes. ......................... 54

4.5 Testing and embedding procedure for \( R \) nodes - Part 1. .................. 55

4.6 Testing and embedding procedure for \( R \) nodes - Part 2. .................. 56

4.7 The c-planarity testing and embedding algorithm for c-connected clustered graphs - Part 1 ................................................................. 57

4.8 The c-planarity testing and embedding algorithm for c-connected clustered graphs - Part 2 ................................. 58

5.1 (a) An example of a cycle with labels in \( \{a, b, c\} \). (b) The cycle with extra edges. (c) The corresponding clustered drawing of the cycle. .......... 59

5.2 The smallest 3-cluster cycle that is not c-planar. (a) The cycle with labels; the dashed lines represent the unique saturator. Note that the cycle and its saturator form a \( K_{3,3} \) graph, and thus the saturator is not planar. (b) The corresponding inclusion tree. ................................. 61
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3</td>
<td>Illustration for the proof of Lemma 17. Circles represent clusters.</td>
</tr>
<tr>
<td>5.4</td>
<td>Zig-zag removal. (a) Starting configuration. (b) Rearrangement of the embedding.</td>
</tr>
<tr>
<td>5.5</td>
<td>(a) The drawing of $x'\alpha'y'$ in $\Gamma_{\tau}$. The gray zones are part of cluster regions. Note that $\tau$ may pass through each cluster many times. (b) After the insertion of path $y'y''$. (c) After the insertion of path $x''\alpha''y''$, the last edge is connected to the first vertex of $\sigma_2$.</td>
</tr>
<tr>
<td>5.6</td>
<td>(a) The drawing of $x'\alpha'y''\alpha''y''$ in $\Gamma_\sigma$. The gray zones are part of cluster regions. Note that $\tau$ may pass through each cluster many times. (b) The drawing of $\Gamma_{\tau}$ after $y'y''\alpha''y''$ was deleted and $P$ was moved.</td>
</tr>
<tr>
<td>5.7</td>
<td>Illustration for the proof of Lemma 18.</td>
</tr>
<tr>
<td>5.8</td>
<td>Illustration for the proof of Lemma 19. Triangular and square vertices show the subdivision of $K_{3,3}$.</td>
</tr>
<tr>
<td>5.9</td>
<td>The construction of a c-planar drawing for a cycle $\sigma$ when $\text{Balance}(\sigma) = 0$ (a) and when $\text{Balance}(\sigma) = 3$ (b).</td>
</tr>
<tr>
<td>5.10</td>
<td>Illustration for the proof of Lemma 20.</td>
</tr>
<tr>
<td>5.11</td>
<td>The construction of a c-planar drawing of a 3-cluster cycle $\sigma$ in the case in which $\text{Balance}(\sigma) = 3$.</td>
</tr>
<tr>
<td>5.12</td>
<td>A clustered graph where at each level of the inclusion tree the nodes form a cycle. (a) A c-planar drawing. (b) The inclusion tree augmented with dashed edges that show the adjacencies between nodes at the same level.</td>
</tr>
<tr>
<td>5.13</td>
<td>(a) Drawing $\Gamma_{G'}$ of $G'$ with edges of the planar saturator added to the external or internal face. (b) Drawing $\Gamma_{C'}$ in which two faces (called the internal and external face) touching all the clusters can be found. (c) saturator edges joining suitable vertices of $\Gamma_{C'}$.</td>
</tr>
<tr>
<td>5.14</td>
<td>(a) Examples of cycles which can be drawn without crossings ((a) and (c)) and which cannot ((b) and (d)).</td>
</tr>
<tr>
<td>5.15</td>
<td>A fountain cluster.</td>
</tr>
<tr>
<td>5.16</td>
<td>An example of cluster expansion: (a) A non-fountain cluster $\mu$. (b) The result of the cluster expansion.</td>
</tr>
<tr>
<td>5.17</td>
<td>The result of a feasible (a) and a non-feasible (b) cluster expansion.</td>
</tr>
<tr>
<td>5.18</td>
<td>An example of pipe contraction: (a) pipe $b$ before contraction; (b) The result of the contraction of $b$.</td>
</tr>
<tr>
<td>5.19</td>
<td>A drawing $\Gamma'$ of $C'$ (a) and the corresponding drawing $\Gamma$ of $C$ (b).</td>
</tr>
<tr>
<td>5.20</td>
<td>Three cases used in the proof of Lemma 27: (a) cluster $\mu_{i+1}$ has degree different from two; (b) $\mu_{i+1}$ has degree two and $b_i$ is the only base for $\mu_{i+1}$; and (c) $\mu_{i+1}$ has degree two and has two bases.</td>
</tr>
</tbody>
</table>
6.8 A c-planar drawing of clustered cycle C whose graph of the clusters $G^1(C)$ is a path. ............................................................... 93
6.9 Algorithm ClusteredCyclePlanarityTesting .................................. 94

7.1 A screenshot of the BGPlay system showing the same AS-graph shown in Fig. 7.2. ................................................................. 99
7.2 A screenshot of the BGPlay system enhanced by the topographic map approach described in this chapter. The AS that originates the prefix is 137, highlighted in red and indicated by an arrow. Our approach clearly shows the importance of the ASes traversed by paths ending into 137. ... 100
7.3 Examples of a real topographic map (courtesy of the U.S. GS). .......... 101
7.4 An handmade layered drawing showing the drawbacks of this kind of representation. Observe how paths zigzag while climbing and descending the hierarchy. Also, ASes that are very far in the network may be represented by nodes which are very near in the drawing (as the black and the gray nodes). .............................................................. 105
7.5 Example of application of the algorithm. Phase 1, the addition of the first fence. ................................................................. 108
7.6 Example of application of the algorithm. Phase 1, the addition of the second fence. ............................................................... 109
7.7 Example of application of the algorithm. Phase 1, the addition of the fourth fence. ................................................................. 110
7.8 Example of application of the algorithm. Final layout after Phase 2. The actual picture shown to the user is presented in Fig. 7.2. ................. 111
7.9 Density of EEX (normalized number of edge crossings). CIRCLES and RIGID FENCES are more affected by edge crossings. ................. 114
7.10 Density of NDR (node distance ratio). LOOSE FENCES and SOFT FENCES show a better “resolution rule” for their nodes. ................. 115
7.11 Density of ELSD (normalized edge-length standard deviation). LOOSE FENCES and SOFT FENCES show more homogeneous edge lengths. ... 116
7.12 Density of $CI_1$ (coastline indention of region 1). Augmenting the mobility factor progressively increases the indention of the border of region 1. ... 117
7.13 Density of $CI_2$ (coastline indention of region 2). LOOSE FENCES and SOFT FENCES have a comparable quality with respect of this measure. CIRCLES and RIGID FENCES produce, obviously, smoother borders. ... 118
7.14 Density of $DV_1$ (density variation between areas $A_1$ and $A_2$). A more homogeneous density is shown by LOOSE FENCES and SOFT FENCES. ... 119
7.15 The map of the AS-graph with highest density in the test suite (Algorithm SOFT FENCES). .............................................................. 120
7.16 Three topographic maps of the same AS-graph. They were produced by different algorithms: (a) **CIRCLES**, (b) **RIGID FENCES**, and (c) **SOFT FENCES**. .......................................................... 120

1 A 3-cluster graph that is not c-planar. The underlying graph is a tree. . . . 122
2 An example that shows that the c-planarity for cycles is only a necessary condition but not a sufficient one for the c-planarity of more complex graphs. The graph of the clusters is supposed to have fixed embedding while the underlying graph is planar and composed by three paths between two vertices (each path is drawn in the picture with a different line-style). .......................................................... 123
Chapter 1

Introduction

Many applications need to visualize large sets of objects and how these objects are interrelated; the relations between such objects are quite often described by a set of binary relations. Mathematicians and computer scientists call a set of binary relations a graph, where the objects of the relations are called vertices and the relations are called edges.

In many graphs is possible to find semantic relations among vertices, that allow to group them into clusters. Clusters of vertices may also be artificially introduced with the purpose of making easy the navigation of large graphs, allowing representations at different levels of abstraction (see e.g. [DGK03]).

This kind of graphs are effectively modeled using clustered graphs [FCE95b]. In a clustered graph vertices are grouped into clusters, which in turn can be grouped into other clusters; the inclusion relation among clusters generates a hierarchy. The graph without considering the clusters is usually called the underlying graph, while the hierarchy of the clusters is the inclusion tree.

Drawing a clustered graph is required in many applications. To give a few examples:

**Networking** The BGP protocol groups local area networks and routers into Autonomous Systems. In the Autonomous Systems that use the OSPF protocol routers and networks are, in turn, clustered into Areas.

**Information Systems** In process analysis, processes are clustered into other processes at different abstraction levels. Also, in data analysis, the entities of an Entity-Relationship schema are often grouped into clusters of entities with similar features.
1. Introduction

Social Networks The actors of a social network are typically grouped into classes of actors with affine features. This process is often repeated and originates a hierarchy of classes.

When representing a network containing clusters, it is quite common to draw the elements of each cluster inside the same region of the plane. Also, disjoint clusters usually lie into disjoint regions.

A common aesthetic criterion in graph drawing is the planarity of the graph. This criterion can be also applied to clustered graphs: also in this case we want to produce planar drawings of the graphs, but with the additional constraint of avoiding unnecessary intersections between the edges of the graph and the border of regions representing the clusters. This kind of drawings are defined as c-planar drawings. The clustered planarity (c-planarity) field studies the interplay between the classical planarity of graphs and the presence of clusters. For its practical interest and because of its theoretical appeal, clustered planarity is attracting increasing attention from many researchers. The purpose of this thesis is to study in depth the problem of c-planarity testing for clustered graphs. The problem of stating the computational complexity of testing the c-planarity of a clustered graph is indeed still open, and the current polynomial time algorithms only deal with specific classes of graphs. Namely, as far as we know, the problem can be solved in polynomial time only if each cluster induces a connected subgraph (in this case the graph is c-connected), or if the graph belongs to a specific family.

The first contribution of this thesis is a structural characterization of c-planarity for c-connected clustered graphs whose underlying graph is biconnected. The characterization is based on the interplay between the hierarchy of the clusters and the hierarchy of the triconnected components of the underlying graph $G$. It is given in terms of properties of the skeletons of the nodes of the SPQR-tree of $G$. Notice that in at least three other papers [Dah98, DKM06, Len89] the relationship between triconnectivity and c-planarity has already been studied. We also easily extend the characterization to general clustered graphs exploiting the decomposition of $G$ into its biconnected components.

Further, based on our c-planarity characterization, we present a new linear time c-planarity testing and embedding algorithm for c-connected clustered graphs. Such an algorithm is reasonably easy to implement, since it is based on simple algorithmic tools as the computation of lowest common ancestors, minimum and maximum spanning trees, and bucket sorts. It also makes use of well-established data structures as SPQR-trees and BC-trees [DT96, GM01] (both in their simple static version).
the test fails, our algorithm identifies a structural element responsible for the non-c-planarity of the input clustered graph.

We are also interested in exploring classes of graphs which are inherently non-connected in order to investigating the impact of non-connectivity on the complexity of c-planarity testing. Indeed, most of the algorithms for non-c-connected graphs assume that the clustered graph has several connected clusters, and they use the c-planarity testing for c-connected graph as a subroutine. In this thesis we describe new classes of highly non-connected clustered graphs, and we show how the c-planarity can be tested in linear time. Namely, we introduce the classes of 3-cluster cycles, clustered cycles, and some generalizations of this classes. In all these classes, the underlying graph is a cycle and two consecutive vertices of the cycle belong to different clusters; hence we have that each cluster is disconnected, and it is no possible to apply the test for connected graphs. This thesis illustrates new methodologies and algorithms to handle with these classes of clustered graphs.

Finally, we show an example of application of clustered drawing in the field of network visualization. We introduce a new visualization metaphor for the Internet graph at Autonomous System (AS) level, showing at the same time the connections between the ASes and their collocation in the Internet Hierarchy. This kind of metaphor allows an effective visualization of the AS graph.

This thesis is organized as follows:

- Chapter 2 contains basic definition about graphs and clustered graphs.
- Chapter 3 proposes the new characterization of c-planarity for c-connected clustered graphs.
- Chapter 4 illustrates the c-planarity testing and embedding algorithm for c-connected clustered graph, based on the characterization proposed in Chapter 3.
- In Chapter 5 we introduce simple families of clustered graphs that are highly unconnected, and we show efficient c-planarity testing and drawings algorithms.
- In Chapter 6 we introduce the class of clustered cycles, and we show a polynomial time c-planarity testing and embedding algorithm for this class.
- Chapter 7 proposes an example of application of clustered drawing in the context of network visualization.
- Finally, we present our conclusions and several open problems.
Chapter 2

Background

This chapter contains basic definitions about graphs and clustered graphs. A brief description of the SPQR-tree data structure is also provided.

2.1 Graphs

An undirected graph is a pair \( G = (V, E) \), where \( V \) is a finite non-empty set of elements called vertices or nodes of \( G \), and \( E \) is a finite, possibly empty, set of elements called edges or arcs of \( G \). An edge \( e \in E \) is an unordered pair \((u, v)\) of vertices of \( G \); vertices \( u \) and \( v \) are called end-vertices of \( e \), or simpler \( \) vertices of \( e \). We write \( e(u, v) \) to denote an edge \( e \) with vertices \( u \) and \( v \), and we say that \( e \) connects \( u \) and \( v \). Vertices \( u \) and \( v \) are said to be adjacent, while \( e \) and any of its end-vertices are incident. Two edges are adjacent if they have a common vertex. The number of edges that are incident to a vertex \( v \) is called the degree of \( v \), and is denoted by \( \text{deg}(v) \).

An edge \( e(u, v) \) of a graph \( G \) is a self-loop if \( u = v \). Also, \( G \) contains multiple edges if it has two or more edges with the same end-points. A graph is simple if it has neither self-loops nor multiple edges.

A path \( p \) between two vertices \( v_0 \) and \( v_k \), \( k \geq 1 \), of a graph \( G \) is an alternating sequence \( v_0, e_1, v_1, e_2, \ldots, e_k, v_k \) of vertices and edges of \( G \), where \( e_i = (v_{i-1}, v_i), (i = 1, \ldots, k) \). Vertices \( v_0 \) and \( v_k \) are the end-vertices of \( p \). We often write \( p = (v_0, v_1, \ldots, v_k) \) to denote path \( p \). If all vertices of \( p \) are distinct, \( p \) is said to be a simple path. The length of a path is the number of its edges. If the end-vertices of \( p \) coincide (i.e. \( v_0 = v_k \)) the path is a cycle. A simple cycle is a cycle such that only its end-vertices coincide, while all its other vertices are distinct. A graph is said to be acyclic if it does not contain any cycle. A graph that is not acyclic is called cyclic.
2. Background

A subgraph $G'$ of a graph $G = (V, E)$ is a pair $(V', E')$, where $V' \subseteq V$ and $E' \subseteq E$. In particular, if $V' = V$, subgraph $G'$ is a spanning subgraph of $G$. Also, if $V'$ is a subset of $V$, the subgraph of $G$ induced by $V'$ is the graph $G' = (V', E')$, where $E' \subseteq E$ is the set of all edges of $G$ connecting any two vertices of $G$ that are in $V'$ (see Figure 2.1). Note that the subgraph induced by a subset of vertices is uniquely determined.

![Figure 2.1](image_url) (a) A graph $G$ with 7 vertices. (b) The subgraph of $G$ induced by vertices $v_1, v_2, v_3, v_6$ and $v_7$.

A subdivision of a graph $G = (V, E)$ is a graph $G'$ obtained from $G$ by replacing a subset $E'$ of edges of $G$ with paths. Namely, if $e = (u, v) \in E'$, it is replaced with a path $p = (u = v_0, v_1, \ldots, v_k = v)$, $k > 1$, where the vertices $v_1, \ldots, v_{k-1}$ do not belong to $V$. Subdivision $G'$ is said to be homeomorphic to graph $G$. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there exists a one-to-one correspondence between sets $V_1$ and $V_2$ and a one-to-one correspondence between sets $E_1$ and $E_2$. In practice, two graphs that are isomorphic can be considered as the same object.

A graph is complete if it has an edge connecting every pair of vertices. A complete graph with $n$ vertices is usually denoted as $K_n$. A graph $G$ is bipartite if the set of its vertices can be partitioned into two sets $V_1$ and $V_2$ such that every edge of $G$ connects a vertex in $V_1$ to a vertex in $V_2$. A graph $G$ is complete bipartite if it is bipartite and every vertex in $V_1$ is adjacent to all vertices in $V_2$. A complete bipartite graph such that $|V_1| = n_1$ and $|V_2| = n_2$ is usually denoted as $K_{n_1,n_2}$.

A directed graph $G$ is defined similarly to a graph, except that every edge $e$ of $G$ is now an ordered pair $(u, v)$ of vertices of $G$; vertices $u$ and $v$ are still called end-vertices of $e$, or simpler vertices of $e$. We also say that $e = (u, v)$ leaves vertex $u$ and enters
vertex \(v\). An edge that leaves (enters) a vertex \(v\) is an outgoing (incoming) edge of \(v\), and it is directed from \(u\) to \(v\).

### 2.2 Connectivity

The definitions of this section and of Section 2.3 are given for graphs. In the case of a digraph they are referred to the corresponding undirected graph.

A connected component \(G'\) of a graph \(G\) is a maximal subgraph of \(G\) such that for every pair \(\{u, v\}\) of vertices of \(G'\) there is a path between \(u\) and \(v\) in \(G'\). A graph that has exactly one connected component is said to be connected (or 1-connected). A separating \(k\)-set, \(k \geq 1\), of a graph \(G\) is a set of \(k\) vertices whose removal increases the number of connected components of \(G\). Separating 1-sets and 2-sets are called cutvertices and separation pairs, respectively. A connected graph is biconnected (or 2-connected) if it has no cutvertices. A biconnected graph is triconnected (or 3-connected) if it has no separation pairs. More in general, a graph is \(k\)-connected, \(k \geq 2\), if it has no separating \((k - 1)\)-sets. Obviously, if a graph is \(k\)-connected, it is also \((k - 1)\)-connected. In Figure 2.2 examples of connected, biconnected, and triconnected graphs are given.

![Figure 2.2](image)

Figure 2.2: (a) A connected graph that is not biconnected; \(u\) and \(v\) are two distinct cutvertices. (b) A biconnected graph that is not triconnected; \(\{v, w\}\) and \(\{v, z\}\) are two distinct separation pairs. (c) A triconnected graph.

Observe that, if \(v\) is a vertex of a graph \(G\) the removal of all its adjacent vertices causes \(v\) to become disconnected from the rest of the graph. Thus, the following
2. Background

property holds.

**Property 1** For each vertex $v$ of a $k$-connected graph, $\text{deg}(v) \geq k$.

A graph that is connected and acyclic is called *tree*. A collection of trees (i.e., an acyclic graph) is a *forest*. We refer to the vertices of a tree as *nodes*, to distinguish them from the vertices of a generic graph.

A *rooted tree* is a tree $T$ in which a node is chosen as *root*. Such a choice immediately induces a hierarchy on the set of all nodes of $T$. Namely, with each node of $T$ we can associate an integer number called *depth* (or *level*) of the node. The depth of the root is 0. For any node $\mu$, distinct from the root, the depth of $\mu$ is the length of the unique path between the root and $\mu$ in $T$; every node $\beta \neq \mu$ on this path is an *ancestor* of $\mu$, and $\mu$ is a *descendant* of $\beta$. Also, if $\langle \beta, \mu \rangle$ is an edge of the path between the root and node $\mu$, $\beta$ is the *parent* of $\mu$, and $\mu$ is a *child* of $\beta$. A node without children is a *leaf*. The *depth* of $T$ is the maximum depth of a leaf of $T$. Figure 2.3 shows two examples of trees.

![Diagram of a tree and a rooted tree](image)

*Figure 2.3: (a) A tree. (b) A rooted tree with depth 3; node $\rho$ is the root of the tree; node $\mu$ is a child of $\beta$; the leaves are the white nodes.*

Let $T$ be a rooted tree. If an ordering of the children of every node of $T$ is given, we say that $T$ is an *ordered rooted tree*.

Let $G$ be a connected graph. A *biconnected component* (or *block*) of $G$ is a maximal biconnected subgraph of $G$. Observe that any cutvertex of $G$ belongs to at least two distinct blocks. It is possible to construct a tree $T$, called *block-cutvertex tree*.
Connectivity

that describes the set of all blocks of $G$ and the relationships among them. This tree, introduced by F. Harary [Har72], can be computed in linear time. It is defined as follows.

- With each block of $G$ there is an associated $B$-node of $T$.
- With each cutvertex of $G$ there is an associated $C$-node of $T$.
- There is an edge in $T$ connecting a $B$-node $\beta$ to a $C$-node $\chi$, if and only if the cutvertex associated with $\chi$ belongs to the block associated with $\beta$.

Observe that in a block-cutvertex tree there are neither edges connecting two $B$-nodes nor edges connecting two $C$-nodes. In Figure 2.4 it is shown a graph and the corresponding block cutvertex tree.

![Figure 2.4](image_url)

Figure 2.4: (a) A graph and its blocks; each block is bounded by a dashed region; the numbered light grey vertices are cutvertices. (b) The block cutvertex tree of the graph in (a); the $B$-nodes are the big circles.

A Depth First Search (DFS in short) is a technique for visiting all the vertices of a graph ([Tar72]). It starts from an arbitrary vertex, called the root, and carries on moving from the current vertex to an adjacent one until unexplored adjacent vertices can be found. When all adjacent vertices of the current vertex are explored, the traversal backtracks to the first vertex that has still unexplored adjacent vertices.
2. BACKGROUND

Each vertex \( v \) is assigned a \textit{DFS index}, denoted \( \text{DFS}(v) \), which specifies the order in which it was reached by the DFS visit, starting from the root \( r \) which has \( \text{DFS}(r) = 1 \).

The edges used by the DFS visit to move from one vertex to the next one are called \textit{tree-edges} and form a spanning tree of \( G \), called the \textit{DFS tree}. The remaining edges are \textit{back-edges}. The \textit{ancestors} of a vertex \( v \) are the vertices in the unique chain of tree-edges from \( v \) to the root \( r \). If a vertex \( u \) is an ancestor of \( v \), then \( v \) is a \textit{descendant} of \( u \). A tree-edge links a \textit{parent} vertex to a \textit{child} vertex, the former (latter) being the one with lowest (highest) DFS-index, while a back-edge is thought to be oriented exiting the descendant and entering the ancestor.

2.3 Drawings and Planarity

Let \( G \) be a graph. A plane \textit{drawing} of \( G \) maps each vertex of \( G \) into a point of the plane, and each edge \((u,v)\) of \( G \) into a simple Jordan curve between the two points corresponding to \( u \) and \( v \), respectively.

A drawing is \textit{planar} if no two distinct edges intersect. A graph is \textit{planar} if it admits a planar drawing. Planar graphs play a crucial role in the graph drawing field and in the whole graph theory [DETT99, NC88].

A planar drawing \( \Gamma \) subdivides the plane into topologically connected regions called \textit{faces}. Exactly one of these faces is an unbounded region, and it is called \textit{external face}. All the other faces are said to be \textit{internal}. In Figure 2.5 it is shown an example of a planar drawing of a planar graph: \( f \) and \( g \) are two internal faces; face \( h \) is the external one. The \textit{degree} of a face \( f \), denoted as \( \text{deg}(f) \), is the number of edges encountered while walking on the border of \( f \) clockwise. In particular, if the graph is biconnected \( \text{deg}(f) \) always coincides with the number of edges that belong to the border of \( f \). In the rest of this work, in order to simplify the notation, we often speak of edges of a face \( f \) to mean the edges that belong to the border of \( f \). In this way, we can also consider a face as a cycle, and describe it as a circular sequence of vertices and edges.

There is a simple necessary condition for connected planar graphs, known as the Euler’s formula.

**Theorem 1 (Euler 1750)** Let \( G = (V,E) \) be a planar graph and let \( F \) be any set of faces of a planar drawing of \( G \). Then:

\[
|E| = |V| + |F| + 2.
\]

Euler’s formula allows to assert that the number of faces of any planar drawing of a planar graph \( G \) does not depend on the choice of the drawing itself.
An important result due to Kuratowski provides a characterization of the set of planar graphs.

**Theorem 2 (Kuratowski 1930) [Kur30]** A graph is planar if and only if it does not contain any subdivision of $K_5$ and $K_{3,3}$.

It is easy to prove that a graph is planar if and only if all its connected components are planar; further a connected graph is planar if and only if all its blocks are planar.

We now observe that a planar drawing $\Gamma$ of a planar graph $G$ induces a circular clockwise ordering of the edges incident on each vertex. Two planar drawings $\Gamma_1$ and $\Gamma_2$ of $G$ are equivalent if they induce the same circular clockwise ordering of edges around vertices, and if they have the same external face. Such binary relationship is clearly an equivalence relationship. We call planar embedding of $G$ an equivalence class of planar drawings of $G$. A planar graph $G$ with an associated embedding $\phi$ is an embedded planar graph, and it is often denoted as $G_\phi$. Note that all the drawings in the embedding $\phi$ have the same set of faces and the same external face. For this reason, we can speak without ambiguity of faces of $G_\phi$. To specify that a drawing $\Gamma$ belongs to an embedding $\phi$ of $G$, we can say that $\Gamma$ preserves $\phi$, or that $\Gamma$ is a drawing of $G_\phi$. In Figure 2.6 three different drawings of a planar graph are shown. The drawings in Figure 2.6 (a) and Figure 2.6 (b) are equivalent.

In order to simplify the terminology, since we mainly work with planar graphs, we often use the term embedding instead of planar embedding.
2. BACKGROUND

![Three planar drawings of the same graph. The drawings in (a) and (b) are equivalent.](image)

Let $G_\phi$ be an embedded planar graph. The dual graph $G^*$ of $G_\phi$ is a graph that has a vertex for each face of $G_\phi$, and an edge $(f, g)$ between two faces $f$ and $g$ (not necessarily distinct) for each edge of $G_\phi$ that is shared by $f$ and $g$. Figure 2.7 shows an embedded planar graph and its dual. Observe that the embedding $\phi^*$ of $G^*$ can be derived from the embedding $\phi$ of $G$. We also remark that, in general, $G^*$ can have self-loops and multiple edges, even if $G$ is a simple graph. Further, two embedded planar graphs with the same embedding have the same dual graph.

Many algorithms in graph drawing are specific for planar graphs. Testing if a graph is planar can be done in linear-time [HT74, BL76, ET76, dFR82, LEC67]. Planarity testing algorithms can be also modified to determine a planar embedding if the graph is found to be planar [CNAO85, MM96].

A further interesting algorithm is based on a characterization given by de Fraysseix and Rosenstiehl [dR85]. This algorithm has not been fully described in the literature but has a very efficient implementation in the Pigale software library [dO].

However, although the planarity problem has been carefully studied in the above cited literature, the story of the planarity testing algorithms enumerates several more recent contributions. The motivations behind such relatively new papers are two-fold. On one side, even if the known algorithms are combinatorially elegant, they are quite difficult to understand and to implement. On the other side, the researchers are interested in deepening the relationships between planarity and Depth First Search (DFS). Such relationships are clearly strong but, probably, up to now, not completely understood.
Two recent DFS-based planarity testing algorithms, whose similarities were stressed in [Tho99], are those presented by Shih and Hsu [SH93, SH99, Hsu03, Boy05] and by Boyer and Myrvold [BM99, BCPD03, BCPD04, BM04].

When a graph is not planar, a pre-processing step can be executed in which the graph is made planar by adding a suitable set of dummy vertices to replace crossings. Such a step is usually called planarization. Reducing the number of dummy vertices (i.e., of crossings) added by a planarization algorithm is an important target for obtaining more readable drawings. For a survey on planarization heuristic algorithms see [DETT99]. Finding the minimum number of crossings and finding a maximum planar subgraph are both NP-hard problems. Combinatorial optimization techniques for the maximum planar subgraph problem have been investigated in [JM96].

For more detail about planarity and connectivity of graphs see [Eve79].

### 2.4 SPQR-trees

A *split pair* \( \{u, v\} \) of a graph \( G \) is either a separation pair or a pair of adjacent vertices. A *maximal split component* of \( G \) with respect to a split pair \( \{u, v\} \) (or, simpler, a
maximal split component of \( \{u, v\} \) is either an edge \( (u, v) \) or a maximal subgraph \( G' \) of \( G \) such that \( G' \) contains \( u \) and \( v \) and \( \{u, v\} \) is not a split pair of \( G' \). A vertex \( v \) distinct from \( u \) and \( v \) belongs to exactly one maximal split component of \( \{u, v\} \). We call the split component of \( \{u, v\} \) a connected subgraph of \( G \) that is the union of any number of maximal split components of \( \{u, v\} \).

In the following, we summarize SPQR-trees. For more details, see [DT96]. An example of an SPQR-tree is shown in Fig. 2.8.

SPQR-trees are closely related to the classical decomposition of biconnected graphs into triconnected components. Let \( \{s, t\} \) be a split pair of \( G \). A maximal split pair \( \{u, v\} \) of \( G \) with respect to \( \{s, t\} \) is a split pair of \( G \) distinct from \( \{s, t\} \) such that, for any other split pair \( \{u', v'\} \) of \( G \), there exists a split component of \( \{u', v'\} \) containing vertices \( u, v, s, \) and \( t \). Let \( e = (s, t) \) be an edge of \( G \), called reference edge. The SPQR-tree \( T \) of \( G \) with respect to \( e \) describes a recursive decomposition of \( G \) induced by its split pairs. Tree \( T \) is a rooted ordered tree whose nodes are of four types: S, P, Q, and R. Denote by \( G' \) the st-biconnected graph obtained from \( G \) by removing \( e \). Each node \( \mu \) of \( T \) has an associated st-biconnected multigraph, called the skeleton of \( \mu \) and denoted by \( \text{skeleton}(\mu) \). Also, it is associated with an edge of the skeleton of the parent \( v \) of \( \mu \), called the virtual edge of \( \mu \) in \( \text{skeleton}(v) \). Tree \( T \) is recursively defined as follows.

**Trivial Case:** If \( G \) consists of exactly one edge between \( s \) and \( t \), then \( T \) consists of a single Q-node whose skeleton is \( G \) itself. See for example vertices 1 and 4 in Fig. 2.8.

**Parallel Case:** If the split pair \( \{s, t\} \) contains at least two maximal split components \( G_1, \ldots, G_k, k \geq 2 \), the root of \( T \) is a P-node \( \mu \). Graph \( \text{skeleton}(\mu) \) consists of \( k \) parallel edges between \( s \) and \( t \), denoted \( e_1, \ldots, e_k \). See for example vertices 3 and 9 in Fig. 2.8.

**Series Case:** If the split pair \( \{s, t\} \) has exactly one maximal split component \( G' \) which is not a single edge and if \( G' \) has cutvertices \( c_1, \ldots, c_{k-1}, k \geq 2 \) in this order on a path from \( s \) to \( t \), the root of \( T \) is an S-node \( \mu \). Graph \( \text{skeleton}(\mu) \) is the path \( e_1, \ldots, e_k \), where \( e_i \) connects \( c_{i-1} \) with \( c_i \) (\( i = 2 \ldots k - 1 \)), \( e_1 \) connects \( s \) with \( c_1 \), and \( e_k \) connects \( c_{k-1} \) with \( t \). See for example vertices 5 and 7 in Fig. 2.8.

**Rigid Case:** If none of the above cases applies, let \( \{s_1, t_1\}, \ldots, \{s_k, t_k\} \) be the maximal split pairs of \( G \) with respect to \( \{s, t\} \) (\( k \geq 1 \) and, for \( i = 1, \ldots, k \), let \( G_i \) be the union of all the maximal split components of \( \{s_i, t_i\} \). The root of \( T \) is an R-node \( \mu \). Graph \( \text{skeleton}(\mu) \) is the triconnected graph obtained from \( G \) by replacing each subgraph \( G_i \) with the edge \( e_i \) between \( s_i \) and \( t_i \). See for example vertices 3 and 8 in Fig. 2.8.
Except for the trivial case, $\mu$ has children $\mu_1, \ldots, \mu_k$, in this order, such that $\mu_i$ is the root of the SPQR-tree of graph $G_i \cup (u_i, v_i)$ with respect to reference edge $(u_i, v_i)$ $(i = 1, \ldots, k)$. Edge $(u_i, v_i)$ is said to be the virtual edge of node $\mu_i$ in skeleton($\mu$) and of node $\mu$ in skeleton($\mu_i$). Graph $G_i$ is called the pertinent graph of node $\mu_i$, and of edge $(u_i, v_i)$ and it is denoted by pertinent($u_i, v_i$). Vertices $u$ and $v$ are the poles of $G_i$.

The tree $T$ so obtained has a Q-node associated with each edge of $G$, except the reference edge $e$. We complete the SPQR-tree $T$ by adding another Q-node, representing the reference edge $e$, and making it the parent of $\mu$ so that it becomes the root of $T$ (see Fig. 2.8.b for an example).

The SPQR-tree $T$ of a graph $G$ with $n$ vertices and $m$ edges has $m$ Q-nodes and $O(n)$ S-, P-, and R-nodes. Also, the total number of vertices of the skeletons stored at the nodes of $T$ is $O(n)$.

A biconnected graph $G$ is planar if and only if the skeletons of all the nodes of the SPQR-tree $T$ of $G$ are planar. An SPQR-tree $T$ rooted at a given Q-node can be used to represent all the planar embeddings of $G$ having the reference edge (associated with the Q-node at the root) on the external face. Namely, any embedding can be obtained by selecting one of the two possible flips of each skeleton around its poles and selecting a permutation of the children of each P-node with respect to their common poles.

### 2.5 Clustered Graphs

A cluster of a graph is a non empty subset of vertices. A clustered graph $C(G,T)$ is a graph $G$ plus a rooted tree $T$ such that the leaves of $T$ are the vertices of $G$ (see Figs. 2.9 and 2.10 for examples). Each internal node $v$ of $T$ corresponds to the cluster $V(v)$ of $G$ whose vertices are the leaves of the subtree rooted at $v$. The subgraph of $G$ induced by $V(v)$ is denoted as $G(v)$. An edge $e$ between a vertex of $V(v)$ and a vertex of $V \setminus V(v)$ is said to be incident on $v$. Graph $G$ and tree $T$ are called underlying graph and inclusion tree, respectively. For example, Figs. 2.10.b and 2.9.b show the inclusion tree of the clustered graphs represented in Figs. 2.10.a and 2.9.a, respectively.

In a drawing of a clustered graph vertices and edges of $G$ are drawn as points and curves as usual [DETT99], and each node $v$ of $T$ is a simple finite open region $R(v)$ such that:

- $R(v)$ contains the drawing of $G(v)$;
- $R(v)$ contains the region $R(\mu)$ if and only if $\mu$ is a descendant of $v$ in $T$; and
2. BACKGROUND

- any two regions $R(v_1)$ and $R(v_2)$ do not intersect if $v_1$ is not a descendant or an ancestor of $v_2$.

See Figs. 2.9.a and 2.10.a for examples of drawings of clustered graph. We say that edge $e$ and region $R$ have an edge-region crossing if $e$ crosses the boundary of $R$ more than once. Since an edge $e$ that is not incident on a node $v$ crosses $R(v)$ an even number of times, an edge-region crossing implies that:

1. edge $e$ is incident on $v$ and $e$ crosses the boundary of $R(v)$ more than once; or

2. edge $e$ is not incident on $v$ and $e$ crosses the boundary of $R(v)$.

A drawing of a clustered graph is c-planar if it does not have edge crossings and edge-region crossings. For example, the drawing of Fig. 2.9.a is c-planar. A clustered graph is c-planar if it has a c-planar drawing. The complexity of the c-planarity testing problem is still unknown.

2.6 State of the Art of C-Planarity

Connectivity plays an important role in the theory of planar graphs. Namely, a graph is planar if and only if all its connected components are planar. For clustered graphs things change. See, for example, Fig. 2.11. A clustered graph is connected if for each node $v$ of $T$ we have that $G(v)$ is connected. For example, the clustered graph in Fig. 2.9 is connected while the one of Fig. 2.10.a is non-connected since $\mu_2$ and $\mu_7$ are non-connected (graphs $G(\mu_2)$ and $G(\mu_7)$ are composed by two isolated vertices).

An important reference point in the literature on clustered planarity is the work by Cohen, Eades, and Feng [Fen97, FCE95b]. In those papers the terminology that is currently adopted is defined and the two following theorems on connected clustered graphs are given.

**Theorem 3** [Fen97, FCE95b] A connected clustered graph $C(G,T)$ is c-planar if and only if $G$ is planar and there exists a planar drawing $D$ of $G$, such that for each node $v$ of $T$, all the vertices and edges of $G - G(v)$ are on the outer face of the drawing of $G(v)$.

**Theorem 4** [Fen97, FCE95b] Given an $n$-vertex connected clustered graph $C(G,T)$, there exists an algorithm to test if $C$ is c-planar and to compute a planar embedding of it in $O(n^2)$ time.
Lengauer [Len89] found a result analogous to the one in [FCE95b, FCE95a], but in a different context. Namely, in that case the clustered graph is specified in terms of a set of graph patterns and in terms of their composition. The time complexity of the algorithm is linear in the size of the input. However, the input size of Lengauer’s algorithm can be quadratic in the size of the represented clustered graph.

Dahlhaus [Dah98] proposed a linear-time algorithm based on the following main ingredients: a decomposition of $G$ into its biconnected and triconnected components, a weight of each cluster proportional to its size, and the above mentioned characterization of c-planar embeddings. The testing algorithm is based on the incremental construction of a certain planar embedding and on a final test that checks whether this embedding is c-planar. A more recent description of this algorithm is presented in [DKM06].

In [DDM01] a planarization algorithm is given for non c-planar connected clustered graphs. In a planarization algorithm a limited number of dummy vertices is added to the graph, each representing a crossing, so that the resulting graph is c-planar.

A clustered graph is completely connected if for each inner node $v$ of $T$ both $G(v)$ and $G - G(v)$ are connected. Observe that the clustered graph of Fig. 2.9 is completely connected. Completely connected clustered graphs have been studied by Cornelsen and Wagner [CW03], that proved the following elegant characterization.

**Theorem 5** [CW03] A completely connected clustered graph is c-planar if and only if its underlying graph is planar.

However, even if c-planarity attracted a lot of research, the general c-planarity testing problem, for non-connected clustered graphs, is still open. A step in this direction is done in the work by Gutwenger et al. [GJL+02, GJL+03] where the following theorem is given.

**Theorem 6** [GJL+02, GJL+03] There exists an $O(n^3)$ time algorithm to test if an $n$-vertex almost connected clustered graph is c-planar.

An almost connected clustered graph is a graph for which all nodes which correspond to the non-connected clusters lie on the same path in $T$ starting at the root of $T$, or graphs in which for each non-connected cluster its parent cluster and all its siblings in $T$ are connected. For example, the clustered graph of Fig. 2.10.a is almost connected since the two non-connected clusters $\mu_2$ and $\mu_7$ have connected parent and siblings.

There are results in the literature, obtained with different purposes, that can be interpreted in terms of c-planarity testing for non-connected graphs. For example,
Biedl et al. [BKM98] study planar graphs where each vertex is assigned to one of two disjoint classes. They show a linear time algorithm to test if one of such graphs has a planar drawing such that the vertices of the two classes are separated by an horizontal line (y-monotone HH-drawing). Of course this can be interpreted as a c-planarity testing of a graph with exactly two clusters (excluding the root) both at the same level.

Biedl et al. [BKM98] also show that a planar bipartite graph has always a y-monotone HH-drawing. The interpretation in terms of c-planarity is obvious: if a clustered graph $C$ has exactly two clusters at the same level, its underlying graph is planar, and the vertices inside a cluster are not linked by any edge, then $C$ is c-planar. Observe that $C$ is “highly non-connected”. The same result can be obtained by exploiting [DLR90], where the interest of the paper is focused on the upward planarity testing. A graph is upward planar if there exist a planar drawing such that for each edge $(v,w)$, we have that $y(v) < y(w)$ (the y coordinate increases when moving from $v$ to $w$).

Goodrich et al. introduced a polynomial-time algorithm for producing planar drawings of extrovert clustered graphs [GLS05], i.e., graphs for which all clusters are connected or extrovert. A cluster $c$ with parent $p$ is extrovert if and only if $p$ is connected and each connected component of $c$ has a vertex with an edge that is incident to a cluster which is external to $p$.

The relationships among all these classes of clustered graphs are illustrated in Fig. 2.12. The green areas correspond to the classes of clustered graphs described in this thesis.

A clustered graph $C(G,T)$ is flat if all the leaves of $T$ have distance two from the root. This implies that all the non-root clusters have depth 1 in $T$. Hence, in a flat clustered graph $C(G,T)$ a graph of the clusters $G^1(C)$ can be identified. Vertices of $G^1(C)$ are the children of the root of $T$ and an edge $(\mu,\nu)$ exists if and only if an edge of $G$ exists incident to both $\mu$ and $\nu$. An example of a flat clustered graph and the corresponding graph of the clusters is shown in Fig 2.13.

A flat clustered graph is equivalently described by a graph $G$ and a labeling of its nodes with labels $\{l_1, l_2, \ldots, l_k\}$, where $k$ is the number of clusters at the lower level. We say that a flat clustered graph is a $k$-cluster graph if such a labeling uses $k$ different labels.

Flat clustered graphs offer a way to deepen our insight into the properties of non-connected c-planar clustered graphs. In fact, by changing the families of the graphs $G$ and $G^1(C)$, c-planarity problems of increasing complexity can be identified. The works in [BKM98, Bie98] by Biedl, Kaufmann, and Mutzel can be interpreted as a linear time c-planarity test for non connected flat clustered graphs with exactly two clusters.

As it can be seen from Fig. 2.12, most families of clustered graphs that admit a
polynomial time c-planarity testing algorithm contain c-connected graphs. In fact, they need a c-planarity testing algorithm for c-connected graphs as a subroutine.

Many papers address the problem of constructing pleasant easily readable drawings of c-planar graphs. For example Eades, Feng, and Lin [EFL97] give the following theorem on convex cluster drawings, where each cluster is a convex polygon.

**Theorem 7** [EFL97] Let $C$ be an $n$-vertex connected c-planar clustered graph. A planar straight line convex cluster drawing of $C$ can be constructed in $O(n^2)$ time.

The above time bound is improved by Nagamochi and Kuroya [NK04] when $T$ is a binary tree and each cluster of $C$ is biconnected.

Multilevel Visualization of Clustered Graphs are studied in [EF97], while the drawings of clustered graphs on an orthogonal grid are studied in [EFN99].
Figure 2.8: A planar graph (a) and the corresponding SPQR-tree (b). The black nodes of the tree represent the Q-nodes, while the dashed lines in the skeletons represent the reference edges.
Figure 2.9: An example of a c-connected clustered graph \( C \). (a) A c-planar drawing of \( C \). (b) The inclusion tree of \( C \).

Figure 2.10: An example of a clustered graph \( C \) which is non-c-connected. (a) A c-planar drawing of \( C \). (b) The inclusion tree of \( C \).
2. **Background**

Figure 2.11: An example of a non c-planar clustered graph.

Figure 2.12: The classes of clustered graphs for which the c-planarity testing problem can be solved in polynomial time. The green areas indicate the classes of clustered graphs described in this thesis.
Figure 2.13: An example of a flat clustered graph $C$ containing 6 clusters (plus the root). (a) A c-planar drawing of $C$. (b) The inclusion tree of $C$. (c) The graph of the clusters of $C$. 
Chapter 3

A Characterization of C-Planarity for C-Connected Graphs

This chapter proposes a new characterization of c-planarity for c-connected clustered graphs \( CDF^{+}06a \). The characterization is based on the interplay between the hierarchy of the clusters and the decomposition of the underlying graph in biconnected and triconnected components.

3.1 Characterization in the Biconnected Case

In this section we characterize the c-planarity of a c-connected clustered graph \( C(G, T) \) when \( G \) is planar and biconnected. First, we introduce some definitions on the cluster hierarchy. Given a connected non-empty subgraph \( G' \) of \( G \), the allocation cluster of \( G' \), denoted by \( ac(G') \) is the lowest common ancestor in \( T \) of the vertices of \( G' \). The allocation cluster represents the lowest cluster containing all the vertices of \( G' \). In the following we refer to the allocation clusters of many special subgraphs of \( G \) like edges, paths, and cycles. For example in Fig. 3.1 the allocation cluster of path 3,5,8 is \( \gamma \). Two clusters \( \alpha \) and \( \beta \) of \( T \) are comparable when they are on the same path from a leaf to the root of \( T \). If \( \alpha \) and \( \beta \) are comparable, the operators \( \prec \), \( \preceq \), and \( \text{max} \) are defined, where \( \alpha \preceq \beta \) (\( \alpha \prec \beta \)) means that \( \alpha \) is an ancestor (proper ancestor) of \( \beta \) and \( \text{max}(\alpha, \beta) \) is the farthest cluster from the root between \( \alpha \) and \( \beta \). The following properties are easy to prove.

Property 2 Given two connected subgraphs \( G' \) and \( G'' \) of \( G \) sharing a vertex, \( ac(G') \) and \( ac(G'') \) are comparable. Also, if \( G'' \subseteq G' \) then \( ac(G') \preceq ac(G'') \).
Property 3 Let $G'$ be a connected subgraph of $G$. There is at least one edge $e \in G'$ such that $ac(e) = ac(G')$.

For example, in Fig. 3.1 the subgraph induced by vertices 3, 5, 8, 7 has allocation cluster equal to $\alpha$ and $ac((3,7)) = \alpha$.

Property 4 There is at least one edge $e \in G$ such that $ac(e)$ is the root of $T$.

Now we relate the concept of allocation cluster, typical of the clusters hierarchy, to the hierarchy of the triconnected components of $G$, represented by the SPQR-tree $T$. A lowest connecting path of a virtual edge $e = (u,v)$ of the skeleton of a node of $T$ is a path between $u$ and $v$ in pertinent($e$) with maximum allocation cluster. Observe that by Property 2 all the paths connecting $u$ and $v$ are pairwise comparable. The lowest connecting cluster of $e$, denoted by lcc($e$) is the allocation cluster of the lowest connecting path of $e$. Fig. 3.2 shows an example of a virtual edge $e$ and its lowest connecting path. The allocation cluster of path $p_1 = (u,x,w,y,v)$ is $\beta$, while the allocation cluster of path $p_2 = (u,x,z,y,v)$ is $\alpha$. Only $p_1$ is a lowest connecting path of $e$, and lcc($e$) = $\beta$. Observe that if $e \in G$ the lowest connecting path of $e$ is the edge itself.

Consider a skeleton of a node $\mu$ of $T$ and a path $p$ composed by virtual edges of the skeleton. The lowest connecting cluster of $p$ is the lowest common ancestor of the lowest connecting clusters of the edges of $p$. To have some intuition on this definition, consider that each edge of $p$ corresponds to a pertinent graph that has a lowest connecting path. Hence, we can see $p$ as “representative” of the concatenation of the lowest connecting paths traversing such pertinent graphs. Each of such paths has an allocation cluster; the lowest connecting cluster is the lowest common ancestor of such allocation clusters. We shall adopt the same definition of lowest connecting cluster also for cycles and faces of skeleton($\mu$). Also, for technical reasons we define the lowest connecting cluster of an external face as the root of the inclusion tree $T$. In Fig. 3.1.c the faces of the skeletons are labelled with the corresponding lowest connecting clusters.

Now we relate the above definitions to the c-planar embeddings. In the following when we refer to a planar embedding of a pertinent graph we always suppose that it has its poles on the external face.

Theorem 8 A planar embedding of a c-connected clustered graph is c-planar iff it does not exist a cycle $c$ that encloses an edge $e$ such that $ac(e) \prec ac(c)$.

Proof. The idea of the proof of necessity is that $ac(e) \prec ac(c)$ implies that the region of the drawing representing $ac(c)$ would enclose the region representing a proper ancestor of $ac(c)$. The sufficiency is proved by observing that if $c$ and $e$ do not exist, then
there exists a drawing such that the enclosure relationships of the regions representing the clusters respect the inclusion tree.

Let $C(G,T)$ be a $c$-connected clustered graph. First, we prove that if a planar embedding $\Gamma$ of $G$ is $c$-planar then every cycle $c$ of $\Gamma$ does not enclose any edge $e$ such that $ac(e) \prec ac(c)$. Suppose that there exist in $\Gamma$ a cycle $c$ and an edge $e$ such that $e$ is enclosed in $c$ and that $ac(e) \prec ac(c)$. It follows that the region $R(\alpha)$ representing the allocation cluster $\alpha$ of $c$ encloses an edge whose allocation cluster $\beta$ is a proper ancestor of $\alpha$. Since $G(\alpha)$ is connected and since $\Gamma$ is planar, then $R(\beta)$ is enclosed in $R(\alpha)$ and so $\Gamma$ is not $c$-planar.

Now, in the hypothesis that $\Gamma$ is a planar embedding of $G$ such that every cycle $c$ of $\Gamma$ does not enclose any edge $e$ with $ac(e) \prec ac(c)$, we prove that $\Gamma$ is $c$-planar. Construct any planar drawing of $G$ with embedding $\Gamma$ and draw any region $R(\gamma)$ representing a cluster $\gamma$ of $T$ so that $R(\gamma)$ “surrounds” $G(\gamma)$, i.e., $R(\gamma)$ contains any vertex and edge of $G(\gamma)$ and does not contain a vertex or an edge of $G - G(\gamma)$. The obtained drawing $D$ of $C$ is a $c$-planar drawing. Namely, since $D$ is a planar drawing then it has no edge crossings; further, the construction of the regions and the $c$-connectivity of $C$ implies that $D$ has no edge-region crossings. It remains to show that $D$ does not contain two regions $R(\alpha)$ and $R(\beta)$, respectively associated to clusters $\alpha$ and $\beta$ of $T$, such that: (i) $R(\alpha)$ encloses $R(\beta)$ and (ii) $\alpha$ is not an ancestor of $\beta$. This is done by showing that if there exist in $D$ regions $R(\alpha)$ and $R(\beta)$ that verify conditions (i) and (ii), then there exists a cycle $c$ of $\Gamma$ that encloses an edge $e$ such that $ac(e) \prec ac(c)$.

By the construction of the regions, there exists in $D$ a cycle $c$ belonging to $G(\alpha)$ that encloses the subgraph $G^* = G(\beta)$. If $\beta$ is a proper ancestor of $\alpha$ then trivially $ac(e) \prec ac(c)$, for some edge $e \in G^*$. Otherwise, consider the subgraph $G'$ of $G$ that is inside $c$ in $D$, where $c \in G'$. By the $c$-connectivity of $C$, $G'$ is connected. Since $\alpha$ is not an ancestor of $\beta$ and since $\beta$ is not a proper ancestor of $\alpha$ then $\alpha$ and $\beta$ are not comparable. This implies that the allocation cluster of $G'$ is a proper ancestor of both $c$ and $G^*$, that is $ac(G') \prec ac(c)$ and $ac(G') \prec ac(G^*)$. By Property 3 there exists at least one edge $e \in G'$ such that $ac(e) = ac(G')$. Edge $e$ can not be part of $c$ and so it is enclosed in $c$. Hence, we obtain $ac(e) = ac(G') \prec ac(c)$, so we derived a contradiction and this concludes the proof.

Given a node $\mu$ of the SPQR-tree $T$ of the underlying graph $G$ of the clustered graph $C(G,T)$, an embedding of $\text{skeleton}(\mu)$ is $c$-planar if every cycle $c$ of $\text{skeleton}(\mu)$ does not enclose an edge $e$ of $\text{skeleton}(\mu)$ with $lcc(e) \prec lcc(c)$. Intuitively, the $c$-planarity of a skeleton is the one of the embedded graph obtained by substituting each edge of the skeleton with its lowest connecting path.

**Lemma 1** If an embedding $\Gamma$ of $\text{skeleton}(\mu)$ is $c$-planar then for any cycle $c$ in $\Gamma$ and for any face $f$ inside $c$ we have that $lcc(c) \leq lcc(f)$. 27
3. A Characterization of C-Planarity for C-Connected Graphs

The proof of Lemma 1 is analogous to the one of Theorem 8.

Given a virtual edge $e = (u, v)$ and a c-planar embedding $\Gamma$ of $\text{pertinent}(e)$, a lowest connecting path $s$ of $e$ separates $\text{pertinent}(e)$ into two embedded subgraphs each containing $s$. We call highest side $\text{hs}(\Gamma, s)$ and lowest side $\text{ls}(\Gamma, s)$ such subgraphs, where $ac(\text{hs}(\Gamma, s)) \leq ac(\text{ls}(\Gamma, s))$. By Property 2 we have:

**Property 5** $ac(\text{hs}(\Gamma, s)) = ac(\text{pertinent}(e))$.

Hence, the value of $ac(\text{hs}(\Gamma, s))$ does not depend on the choice of the c-planar embedding $\Gamma$ and of $s$ and we can define the highest side cluster of $e$, $\text{hs}(e) = ac(\text{pertinent}(e))$.

**Lemma 2** The value of $ac(\text{ls}(\Gamma, s))$ does not depend on the choice of $s$.

**Proof.** Suppose, by contradiction, that there exist two different lowest connecting paths $s_1$ and $s_2$, with $\text{lcc}(s_1) = \text{lcc}(s_2) = \gamma$, such that $\alpha = ac(\text{ls}(\Gamma, s_1)), \beta = ac(\text{ls}(\Gamma, s_2))$, and $\alpha \neq \beta$. By Property 2, we have that $\alpha$ and $\beta$ are comparable. Assume, w.l.o.g, that $\alpha < \beta \leq \gamma$. Then, there exist an edge $e_1$ such that $ac(e_1) = \alpha$. Edge $e_1$ belongs to $\text{ls}(\Gamma, s_1)$ and does not belong to $\text{ls}(\Gamma, s_2)$, then $e_1$ is necessarily enclosed in a simple cycle $c$ composed by subpath $\overline{s_1}$ of $s_1$ and subpath $\overline{s_2}$ of $s_2$ (see Fig. 3.3). By Property 2, $\text{lcc}(\overline{s_1})$ and $\text{lcc}(\overline{s_2})$ are comparable. By construction, $\text{lcc}(e) = \min\{\text{lcc}(\overline{s_1}), \text{lcc}(\overline{s_2})\} \geq \gamma > \alpha$. Hence, by Theorem 8, $\Gamma$ would be a non-c-planar embedding, contradicting the hypothesis.

Due to Lemma 2 we can define the lowest side cluster of $\Gamma$, $\text{lsc}(\Gamma) = ac(\text{ls}(\Gamma, s))$, and the lowest side cluster of $e$, $\text{lsc}(e) = \max_{\Gamma}\{\text{lsc}(\Gamma)\}$. Observe that the definitions of $\text{hsc}(e)$ and of $\text{lsc}(e)$ hold only if $\text{pertinent}(e)$ is c-planar. For technical reasons if $\text{pertinent}(e)$ is not c-planar we define $\text{hsc}(e) = \text{lsc}(e) = \bot$, where $\bot$ is by convention a proper ancestor of any cluster. See Fig. 3.4 for an example. As another example, Fig. 3.1 contains, for each virtual edge $e$ of the represented skeletons, a triple describing $\text{lcc}(e)$, $\text{lsc}(e)$, and $\text{hsc}(e)$, respectively.

**Property 6** For each edge $e$ of the skeleton of a node of $\Gamma$, $\text{hsc}(e) \leq \text{lsc}(e) \leq \text{lcc}(e)$.

**Property 7** Let $c$ be a cycle of virtual edges in a skeleton of a node of $\Gamma$ and let $e$ be an edge of $c$. We have that $\text{lcc}(c)$ is comparable with $\text{lcc}(e)$, with $\text{lsc}(e)$, and with $\text{hsc}(e)$.

**Property 8** Let $e = (u, v)$ be a virtual edge. Suppose that $\text{pertinent}(e)$ is c-planar. Then in any c-planar embedding $\Gamma$ of $\text{pertinent}(e)$ and for any lowest connecting path
s of e there exist two edges \( e_1 \in hs(\Gamma, s) \) and \( e_2 \in ls(\Gamma, s) \) such that \( ac(e_1) = hsc(e) \) and \( ac(e_2) \preceq lsc(e) \). Also, if \( hsc(e) \prec lcc(e) \) then there exists an edge \( e_1 \in hs(\Gamma, s) \) such that \( ac(e_1) = hsc(e) \) and \( e_1 \notin s \). Similarly, if \( lsc(e) \prec lcc(e) \) then there exists \( e_2 \in ls(\Gamma, s) \) such that \( ac(e_2) \preceq lsc(e) \) and \( e_2 \notin s \).

Two comparable virtual edges \( e_1 \) and \( e_2 \) of a skeleton of a node of \( T \) are incompatible when, assuming w.l.o.g. \( lcc(e_1) \preceq lcc(e_2) \), one of the following conditions hold: (i) \( lcc(e_1) \prec lcc(e_2) \) and \( hsc(e_2) \prec lcc(e_1) \); (ii) \( lcc(e_1) = lcc(e_2) \), \( hsc(e_1) \prec lcc(e_1) \), and \( hsc(e_2) \prec lcc(e_2) \).

For example, the skeleton of the P-node shown in Fig. 3.1.c has three virtual edges that are pairwise compatible. Now we can formulate the characterization.

**Theorem 9** Let \( C(G, T) \) be a c-connected clustered graph where \( G \) is planar and biconnected, and let \( T \) be the SPQR tree of \( G \) rooted at an edge whose allocation cluster is the root of \( T \). \( C \) is c-planar if and only if for each node \( \mu \) of \( T \) the following conditions are true:

1. If \( \mu \) is an R node then the embedding of skeleton(\( \mu \)) is c-planar and each edge \( e \) of skeleton(\( \mu \)) is incident to two faces \( f_1 \) and \( f_2 \) such that the lowest connecting cluster of \( f_1 \) is an ancestor of the highest side cluster of \( e \) and the lowest connecting cluster of \( f_2 \) is an ancestor of the lowest side cluster of \( e \).

2. If \( \mu \) is a P node then

   a) it does not exist a set of three edges of skeleton(\( \mu \)) that are pairwise incompatible and

   b) there exists at most one edge \( e^* \) of skeleton(\( \mu \)) such that the lowest side cluster of \( e^* \) is a proper ancestor of the lowest connecting cluster of \( e^* \) and if there exists such \( e^* \) then for each edge \( e \neq e^* \) of skeleton(\( \mu \)) the lowest connecting cluster of \( e \) is an ancestor of the lowest side cluster of \( e^* \).

**Proof of the Sufficiency**

The sufficiency of the conditions of Theorem 9 can be proved by structural induction starting from the leaves of \( T \). For each leaf \( \mu \) (Q node) of \( T \), pertinent(\( \mu \)) is trivially c-planar. We prove the inductive step by considering any non-leaf node \( \mu \) with children \( \mu_1, \mu_2, \ldots, \mu_k \). Namely, we suppose that their pertinent graphs are c-planar and argument that, if the conditions of the theorem are satisfied, then also pertinent(\( \mu \)) is c-planar. This is done by means of two lemmas, one for the \( R \) nodes and the other for the \( P \) nodes. The lemma for the \( R \) nodes is valid for all nodes whose skeleton has
a fixed embedding and so it is formulated in the most general setting. The case of the $S$ nodes is trivial. A special lemma describes the situation of the root of $\mathcal{T}$ and completes the proof of sufficiency.

**Lemma 3** Let $\mu$ be a node of $\mathcal{T}$ and let $\Gamma_\mu$ be a c-planar embedding of skeleton($\mu$). If (i) the pertinent graphs of the children of $\mu$ are c-planar, and (ii) each edge $e$ of skeleton($\mu$) is incident to two faces $f_1$ and $f_2$ of $\Gamma_\mu$ such that lcc($f_1$) $\preceq$ hsc($e$) and lcc($f_2$) $\preceq$ lsc($e$), then pertinent($\mu$) is c-planar.

**Proof.** Let $e_1,e_2,\ldots,e_k$ be the edges of skeleton($\mu$) and consider for each $e_i$, $i = 1,\ldots,k$, a c-planar embedding $\Gamma_i$ of pertinent($e_i$) such that lsc($\Gamma_i$) = lsc($e_i$).

Denoting by $f_{i1}$ and $f_{i2}$ the faces of $\Gamma_\mu$ incident to $e_i$, construct an embedding $\Gamma$ of pertinent($\mu$) by turning each $\Gamma_i$, $i = 1,\ldots,k$, so that hs($\Gamma_i$) is toward $f_{i1}$ and ls($\Gamma_i$, $s_i$) is toward $f_{i2}$ with lcc($f_{i1}$) $\preceq$ hsc($e_i$) and lcc($f_{i2}$) $\preceq$ lsc($e_i$). The second condition of the lemma makes this possible.

By supposing that $\Gamma_\mu$ is c-planar, we show that $\Gamma$ is also c-planar, that is, for any cycle $c$ of $\Gamma$ and any edge $e$ that is enclosed in $c$ we have ac($c$) $\preceq$ ac($e$) (see Theorem 8).

We denote by $e^* = (v_1,v_2)$ the edge of skeleton($\mu$) containing $e$ in pertinent($e^*$) and we denote by $f^*_1$ and $f^*_2$ the faces incident to $e^*$ in $\Gamma_\mu$ for which lcc($f^*_1$) $\preceq$ hsc($e^*$) and lcc($f^*_2$) $\preceq$ lsc($e^*$). Let $\Gamma^*$ be the embedding of pertinent($e^*$) in $\Gamma$. We also denote by $c^*$ the cycle in $\Gamma_\mu$ whose edges $e_i$ contain in $\cup$ pertinent($e_i$) all the edges of $c$.

Edge $e^*$ may be part of $c^*$ or can be enclosed in $c^*$. In both cases at least one of $f^*_1$ and $f^*_2$ is enclosed by $c^*$. If $e^*$ encloses $f^*_1$, we have hsc($e^*$) $\preceq$ ac($e$) by definition of hsc($e^*$), lcc($f^*_1$) $\preceq$ hsc($e^*$) by hypothesis and by the construction of $\Gamma$, lcc($e^*$) $\preceq$ lcc($f^*_1$) by Lemma 1, and ac($c^*$) $\preceq$ lcc($c^*$) by definition of lowest connecting cluster. Hence, ac($c$) $\preceq$ lcc($c^*$) $\preceq$ lcc($f^*_1$) $\preceq$ hsc($e^*$) $\preceq$ ac($e$).

If $c^*$ encloses $f^*_2$ (and does not enclose $f^*_1$), consider a lowest connecting path $s$ of $e^*$ and the path $p \subseteq c$ between $v_1$ and $v_2$ in pertinent($e^*$). If $e \in ls(\Gamma^*,s)$ then lsc($e^*$) $\preceq$ ac($e$) by definition of lsc($e^*$), lcc($f^*_2$) $\preceq$ lsc($e^*$) by hypothesis and by the construction of $\Gamma$, lcc($c^*$) $\preceq$ lcc($f^*_2$) by Lemma 1, and ac($c$) $\preceq$ lcc($c^*$) by definition of lowest connecting cluster, so ac($c$) $\preceq$ lcc($c^*$) $\preceq$ lcc($f^*_2$) $\preceq$ lsc($e^*$) $\preceq$ ac($e$). If $e \in hs(\Gamma^*,s)$ then there exists a simple cycle $\mathcal{C}$ formed by $p$ and $s$, or by a part of them, such that $e$ is enclosed by or is part of $\mathcal{C}$. In both cases the c-planarity of $\Gamma^*$ implies that ac($\mathcal{C}$) $\preceq$ ac($e$). Also, ac($p$) $\preceq$ ac($\mathcal{C}$) (ac($p$) is an ancestor of ac($s$) and of ac($\mathcal{C}$)) and since $p$ is part of $c$ ac($c$) $\preceq$ ac($p$). So ac($c$) $\preceq$ ac($p$) $\preceq$ ac($\mathcal{C}$) $\preceq$ ac($e$). □

**Lemma 4** Let $\mu$ be a $P$ node such that the pertinent graphs of its children are c-planar. Suppose that: (i) it does not exist a set of three edges of skeleton($\mu$) that are
Characterization in the Biconnected Case

pairwise incompatible, (ii) there exists at most one edge $e^*$ of skeleton($\mu$) such that $lsc(e^*) \prec lcc(e^*)$, and (iii) if there exists $e^*$ then each edge $e \neq e^*$ of skeleton($\mu$) is such that $lcc(e) \geq lsc(e^*)$. Then pertinent($\mu$) is c-planar.

Proof. First, observe that all the edges of skeleton($\mu$) are pairwise compatible since they share the poles. Subdivide the edges of skeleton($\mu$) into two ordered sets $I_L = \{l_1, l_2, \ldots, l_p\}$ and $I_R = \{r_1, r_2, \ldots, r_q\}$ such that all the edges in $I_L$ (in $I_R$) are pairwise compatible. This partition exists since skeleton($\mu$) does not contain three pairwise incompatible edges. Also the edges in $I_L$ (in $I_R$) are ordered so that $lcc(l_p) \leq lcc(l_{p-1}) \leq \ldots \leq lcc(l_1)$ ($lcc(r_q) \leq lcc(r_{q-1}) \leq \ldots \leq lcc(r_1)$). If two or more edges $l_i, l_{i+1}, \ldots, l_m$ ($r_j, r_{j+1}, \ldots, r_n$) in $I_L$ (in $I_R$) have the same lowest connecting cluster, they are ordered so that $hsc(l_m) \leq hsc(l_{m-1}) \leq \ldots \leq hsc(l_1)$ ($hsc(r_n) \leq hsc(r_{n-1}) \leq \ldots \leq hsc(r_1)$). If two or more edges $l_i$ ($r_j$) in $I_L$ (in $I_R$) have the same lowest connecting cluster and the same highest side cluster, they are ordered in any way.

Consider the embedding $\Gamma_{\mu}$ of skeleton($\mu$) obtained by placing its edges in the order $I = \{l_p, \ldots, l_1, r_1, \ldots, r_q\}$. We show that skeleton($\mu$) is c-planar. This is done by considering three edges $e_1, e_2, e_3$ that appear in this order in $\Gamma_{\mu}$ and by proving that the lowest connecting cluster of the cycle $c = \{e_1, e_3\}$ is an ancestor of the lowest connecting cluster of $e_2$. If $e_1$ and $e_3$ are both in $I_L$ (in $I_R$), then $lcc(c) = lcc(e_1) \leq lcc(e_2)$ (resp. $lcc(c) = lcc(e_3) \leq lcc(e_2)$). Otherwise $e_1 \in I_L$ and $e_3 \in I_R$. Suppose $e_2 \in I_L$ ($e_2 \in I_R$), $lcc(c) \geq lcc(e_1) \geq lcc(e_2)$ (resp. $lcc(c) \geq lcc(e_3) \geq lcc(e_2)$).

We now show that $\Gamma_{\mu}$ is such that each edge $e$ of skeleton($\mu$) is incident to two faces $f_1$ and $f_2$ such that $lcc(f_1) \geq hsc(e)$ and $lcc(f_2) \leq lsc(e)$. The internal faces of $\Gamma_{\mu}$ consist of two edges that are consecutive in $I$. We denote by $\{e_1, e_2\}$ an internal face of $\Gamma_{\mu}$ between edges $e_1$ and $e_2$. We have that $lcc(\{l_i, l_{i+1}\}) = lcc(l_{i+1})$, for $1 \leq i < p$, and that $lcc(\{r_j, r_{j+1}\}) = lcc(r_{j+1})$, for $1 \leq j < q$. Since $l_i$ and $l_{i+1}$, for $1 \leq i < p$ ($r_j$ and $r_{j+1}$, for $1 \leq j < q$), are compatible, then $lcc(\{l_i, l_{i+1}\}) \leq hsc(l_i)$ ($lcc(\{r_j, r_{j+1}\}) \leq hsc(r_j)$). Also denoting by $f_2$ the external face of $\Gamma_{\mu}$ we clearly have $lcc(f_2) \leq hsc(l_p)$ and $lcc(f_2) \leq hsc(r_q)$. By the second condition on the $P$ nodes there exists at most one edge $e^*$ such that $lsc(e^*) > lcc(e^*)$, and so $lcc(f_2) \leq lsc(e)$ can be violated only for $e = e^*$. By the ordering of the edges in $\Gamma(\mu)$, we have that either $e^* = r_1$ or $e^* = l_l$. W.l.o.g. suppose that $e^* = r_1$. By the third condition on the $P$ nodes $lcc(\{l_1, r_1\}) \leq lcc(l_1) \leq lsc(r_1)$. By Lemma 3 we can conclude that pertinent($\mu$) is c-planar.

\[ \Box \]

Lemma 5 Let $C(G,T)$ be a c-connected clustered graph and let $(u,v)$ be an edge of $G$ such that ac$(u,v)$ is the root of $T$. If $C(G - (u,v), T)$ admits a c-planar embedding with $u$ and $v$ on the external face, then $C$ is c-planar.
3. A Characterization of C-Planarity for C-Connected Graphs

Proof. Since $ac((u,v))$ is the root of $T$, it is easy to see that $(u,v)$ cannot create cycles enclosing an edge whose allocation cluster is a proper ancestor of the allocation cluster of the cycle. Hence, by Theorem 8, $C$ admits a c-planar embedding obtained by adding $(u,v)$ to the one of $C(G - (u,v), T)$.

The above lemma completes the proof of sufficiency.

Proof of the Necessity

The proof of the necessity of Theorem 9 is split into five lemmas, that, considered altogether, constitute a complete proof. Before giving such lemmas, we discuss the following issue: how limiting is the choice of rooting $T$ to a specific edge whose allocation cluster is the root of $T$? The answer is in the following theorem.

Theorem 10 Let $C(G,T)$ be a c-connected c-planar clustered graph and let $\Gamma$ be any c-planar embedding of $C$. Let $e$ be an edge whose allocation cluster is the root of $T$. Change the external face of $\Gamma$ choosing as external any face containing $e$. The resulting planar embedding is still c-planar.

Proof. Let $f$ be the external face of $\Gamma$ and let $\Gamma'$ be the embedding of $C$ derived from $\Gamma$ with external face $g$ containing $e$. By contradiction, suppose that $\Gamma'$ is not c-planar. By Theorem 8, there exists in $\Gamma'$ a cycle $c$ that encloses an edge $e$ such that $ac(e) \prec ac(c)$. Note that, $ac(c)$ cannot be the root of $T$, hence we have also $ac(e) \prec ac(c)$. Consider $c$ and $f$ in $\Gamma'$, either (i) $c$ encloses $f$ and separates $f$ from $g$ or (ii) $c$ does not encloses $f$ and hence $f$ and $g$ are not separated by $c$. In the first case, $\Gamma$ cannot be c-planar since in this embedding $c$ encloses $e$. In the second case, $\Gamma$ cannot be c-planar since in this embedding $c$ encloses $\mathcal{G}$. In both cases we have a contradiction.

Now we start the proof of the necessity. The first two lemmas are aimed to prove the necessity of the conditions on the skeletons of the $R$ nodes.

Lemma 6 Let $\mu$ be an $R$ node of $T$ such that the conditions of Theorem 9 are satisfied for all nodes of the subtree rooted at $\mu$ but for $\mu$ itself. Namely, suppose that the embedding of skeleton($\mu$) with its poles on the external face is not c-planar. Then $C$ is not c-planar.

Proof. Because of the non c-planarity, the unique embedding of skeleton($\mu$) with its poles $u$ and $v$ on the external face contains a cycle $c^*$ that encloses an edge $e^*$ such that $lcc(e^*) \prec lcc(c^*)$. It is possible to find in each embedding of pertinent($\mu$) a cycle $c$ corresponding to $c^*$ by substituting each virtual edge $e^*$ of $c^*$ with a lowest
connecting path of $e^*$. By the definition of lowest connecting path, $ac(e) = lcc(e^*)$. Also, by the definition of $lcc(e^*)$ we have that pertinent$(e^*)$ contains an edge $e$ with $ac(e) = lcc(e^*)$. Since $\mu$ is a R node, in any embedding of pertinent$(\mu)$ $e$ lies inside $c$. So, because of Theorem 8, pertinent$(\mu)$ is not c-planar with $(u,v)$ on the external face and $C$ is not c-planar with the root of $T$ on the external face. By Theorem 10 it follows that $C$ is not c-planar.

Lemma 7 Let $\mu$ be an R node of $T$ such that the conditions of Theorem 9 are satisfied for all nodes of the subtree rooted at $\mu$ but for $\mu$ itself. Namely, suppose that skeleton$(\mu)$ contains an edge $e$ incident to two faces $f_1$ and $f_2$ ($lcc(f_1) \preceq lcc(f_2)$) with $hsc(e) \prec lcc(f_1)$ or $lsc(e) \prec lcc(f_2)$, then $C$ is not c-planar.

Proof. Concatenating the lowest connecting paths of the edges composing $f_1$ ($f_2$), we can find in pertinent$(\mu)$ a cycle $c_1$ ($c_2$) corresponding to $f_1$ ($f_2$) of skeleton$(\mu)$ with $ac(c_1) = lcc(f_1)$ ($ac(c_2) = lcc(f_2)$). Due to Property 8 there exist two edges $e_1$ and $e_2$ of pertinent$(e)$ such that $ac(e_1) = hsc(e)$, $ac(e_2) \preceq lsc(e)$ and that either $e_1$ lies inside $c_1$ and $e_2$ lies inside $c_2$ or $e_1$ lies inside $c_2$ and $e_2$ lies inside $c_1$. In both cases we have a cycle $c$ that contains an edge $e$ such that $ac(e) \prec ac(c)$, so pertinent$(\mu)$ is not c-planar with $(u,v)$ on the external face and $C$ is not c-planar with the root of $T$ on the external face. By Theorem 10 it follows that $C$ is not c-planar. □

The next three lemmas are to prove the necessity of the conditions of Theorem 9 on the skeletons of the P nodes.

Lemma 8 Let $\mu$ be a P node of $T$ such that the conditions of Theorem 9 are satisfied for all nodes of the subtree rooted at $\mu$ but for $\mu$ itself. Namely, suppose that skeleton$(\mu)$ contains three edges $e_1, e_2, e_3$ that are pairwise incompatible, then $C$ is not c-planar.

Proof. Consider any embedding of skeleton$(\mu)$ and suppose, w.l.o.g., that $e_1, e_2, e_3$ are embedded in this order around the poles. Consider a cycle $c$ of pertinent$(\mu)$ composed by a lowest connecting path of $e_1$ and by a lowest connecting path of $e_3$. We have that $lcc(c)$ is the lowest common ancestor of $lcc(e_1)$ and $lcc(e_3)$. By applying Property 8 it is possible to find an edge $e \in$ pertinent$(e_2)$ such that $ac(e) = hsc(e_2)$. Since $e_2$ is incompatible with both $e_1$ and $e_3$ we have $ac(e) = hsc(e_2) \prec lcc(c)$. Hence, pertinent$(\mu)$ is not c-planar with $(u,v)$ on the external face and $C$ is not c-planar with the root of $T$ on the external face. By Theorem 10 it follows that $C$ is not c-planar. □

Lemma 9 Let $\mu$ be a P node of $T$ such that the conditions of Theorem 9 are satisfied for all nodes of the subtree rooted at $\mu$ but for $\mu$ itself. Namely, suppose that
3. A Characterization of C-Planarity for C-Connected Graphs

skeleton(\mu) contains two edges \(e_1^*\) and \(e_2^*\) with \(lsc(e_1^*) \prec lcc(e_1^*)\) and \(lsc(e_2^*) \prec lcc(e_2^*)\), then \(C\) is not c-planar.

**Proof.** Consider a lowest connecting path \(p_1\) of \(e_1^*\) and a lowest connecting path \(p_2\) of \(e_2^*\). Let \(c\) be the cycle obtained by concatenating \(p_1\) and \(p_2\); we have that \(lcc(c)\) is the lowest common ancestor of \(lcc(e_1^*)\) and \(lcc(e_2^*)\). W.l.o.g. we assume that \(lcc(c) = lcc(e_1^*)\). By Property 8 in any embedding of \(pertinent(e_1^*)\) there exist edges \(e_1\) and \(e_2\) that are separated by \(p_1\) and are such that \(ac(e_1) = hsc(e_1^*)\) and \(ac(e_2) \preceq lsc(e_1^*)\). One between \(e_1\) and \(e_2\) is enclosed by \(c\). By the above inequalities \(ac(e_1) \prec lcc(c)\), \(ac(e_2) \prec lcc(c)\), and hence \(pertinent(\mu)\) is not c-planar with \((u, v)\) on the external face and \(C\) is not c-planar with the root of \(T\) on the external face. By Theorem 10 it follows that \(C\) is not c-planar.

**Lemma 10** Let \(\mu\) be a P node of \(T\) such that the conditions of Theorem 9 are satisfied for all nodes of the subtree rooted at \(\mu\) but for \(\mu\) itself. Namely, suppose that \(skeleton(\mu)\) contains an edge \(e^*\) with \(lsc(e^*) \prec lcc(e^*)\) and an edge \(e \neq e^*\) such that \(lsc(e^*) \prec lcc(e)\), then \(C\) is not c-planar.

**Proof.** By Property 6 we have that \(hsc(e^*) \prec lcc(e)\). Consider a lowest connecting path \(p_{e^*}\) of \(e^*\), a lowest connecting path \(p_e\) of \(e\), and the cycle \(c = p_{e^*} \cup p_e\); \(lcc(c)\) is the lowest common ancestor of \(lcc(e^*)\) and \(lcc(e)\). By Property 8 in any embedding of \(pertinent(e^*)\) there exist edges \(e_1\) and \(e_2\) that are separated by \(p_{e^*}\) and are such that \(ac(e_1) = hsc(e^*)\) and \(ac(e_2) \preceq lsc(e^*)\). One between \(e_1\) and \(e_2\) is enclosed by \(c\). By the above inequalities \(ac(e_1) \prec lcc(c)\), \(ac(e_2) \prec lcc(c)\), and hence with \((u, v)\) on the external face and \(C\) is not c-planar with the root of \(T\) on the external face. By Theorem 10 it follows that \(C\) is not c-planar.

The above lemma completes the proof of necessity of Theorem 9.

3.2 Characterization of the C-Planarity of General C-connected Clustered Graphs

In this section we extend the characterization given in Section 3.1 to general c-connected clustered graphs.

**Theorem 11** Let \(C(G,T)\) be a c-connected clustered graph and let \(B\) be the BC-tree of \(G\) rooted at a block \(v\) that contains an edge \(e\) whose allocation cluster is the root of \(T\). \(C\) is c-planar if and only if each block \(\mu\) of \(B\) admits a c-planar embedding \(\Gamma_\mu\) such that the parent cut-vertex of \(\mu\) (if any) is on the external face of \(\Gamma_\mu\) and each
child cut-vertex $\rho_i$ of $\mu$ is incident to a face $f_i$ whose lowest connecting cluster is an ancestor of the allocation cluster of pertinent($\rho_i$).

Proof. First, we show the sufficiency of the conditions. Suppose each block $\mu$ admits an embedding $\Gamma_\mu$ respecting the above conditions. We show how to build a c-planar embedding $\Gamma_G$ of $G$, by suitably merging the embeddings $\Gamma_\mu$. We traverse top-down $\mathcal{B}$ starting at its root $v$. Let $\mu$ be the current block, and let $\mu_1, \mu_2, \ldots, \mu_k$ be the blocks whose parent cutvertices $\rho_1, \rho_2, \ldots, \rho_k$ (not necessarily distinct) are children of $\mu$. We select a face $f_i$ of $\Gamma_\mu$ incident to $\rho_i$ and whose lowest connecting cluster is an ancestor of the allocation cluster of pertinent($\rho_i$). We embed $\Gamma_\mu$ into $f_i$ identifying the two instances of $\rho_i$ in $\mu$ and $\mu_i$. Distinct children of the same cutvertex are embedded in such a way that one does not enclose the other. Now we show that the obtained embedding $\Gamma_G$ is c-planar. Suppose, by contradiction, that $\Gamma_G$ is not c-planar. Let $p$ be a simple cycle enclosing an edge $e$ with $ac(e) \prec ac(p)$. Observe that, since $p$ is simple, all edges of $p$ are contained into the same block $\mu^*$. If $e$ also belongs to $\mu^*$, then $\Gamma_{\mu^*}$ is not c-planar, contradicting the hypothesis. Otherwise, suppose $e$ belongs to pertinent($\rho_i$), where $\rho_i$ is a child cutvertex of $\mu^*$. Hence, $ac(pertinent(\rho_i)) \leq ac(e)$. By construction, $ac(f_i) \preceq ac(pertinent(\rho_i))$. Since each edge of $f_i$ belongs or is internal to $p$, Theorem 8 ensures that $ac(p) \preceq ac(f_i)$. Therefore, we have $ac(p) \preceq ac(e)$ contradicting the hypothesis that $ac(e) \prec ac(p)$.

It is trivial that the c-planarity of the blocks of the BC-tree is a necessary condition for the c-planarity of $C$. Also, by Theorem 10, the choice of rooting $\mathcal{B}$ to a node $v$ containing an edge $e$ whose allocation cluster is the root of $T$, and the choice of rooting $T_v$ to $e$ are not limiting, that is if a c-planar embedding $\Gamma_G$ of $G$ exists, then there exists also a c-planar embedding of $G$ with $e$ on the external face. This leads to assume that $\Gamma_G$ has $e$ on the external face. In order to show the necessity of the other two conditions, suppose that a block $\mu^* \neq v$ does not admit a c-planar embedding with its parent cutvertex on the external face. Then the blocks that are ancestors of $\mu^*$ have to be embedded inside $\mu^*$ and so $e$ cannot be on the external face of $\Gamma_G$, contradicting the hypothesis. Now suppose that a child cutvertex $\rho_i$ of $\mu$ is such that $ac(pertinent(\rho_i)) \prec ac(f_j)$ for each face $f_j$ of $\Gamma_\mu$ incident to $\rho_i$. By Property 3 pertinent($\rho_i$) contains an edge $e_i$ such that $ac(e_i) = ac(pertinent(\rho_i))$. So embedding pertinent($\rho_i$) inside $\mu$ creates a cycle $f_j$ containing an edge $e_i$ such that $ac(e_i) \prec ac(f_j)$, while embedding $\mu$ inside any embedding of pertinent($\rho_i$) implies that $e$ cannot be on the external face of $\Gamma_G$, contradicting the hypothesis. Hence, $C$ is not c-planar with $e$ on the external face and by Theorem 10 it follows that $C$ is not c-planar. \qed
Figure 3.1: Clustered graph. (a) Underlying graph. (b) Inclusion tree. (c) SPQR-tree. The boxes contain the skeletons of selected nodes. The triple on each virtual edge represents $lcc(e)$, $lsc(e)$, and $hsc(e)$, respectively. Faces of skeletons are labeled with their $lcc$. 
Figure 3.2: Lowest connecting paths and clusters. (a) A virtual edge $e = (u, v)$. (b) Graph $\text{pertinent}(e)$; the thick lines show the lowest connecting path $p_1$. (c) Tree $T$ restricted to the clusters in $\text{pertinent}(e)$.

Figure 3.3: Illustration for the proof of Lemma 2. The paths $s_1$ and $s_2$ are drawn in thick lines, while the subpaths $\bar{s}_1$ and $\bar{s}_2$ are drawn in dashed lines.
Figure 3.4: (a) A virtual edge $e = (u, v)$. (b) Graph $\text{pertinent}(e)$. (c) Tree $T$ restricted to the clusters in $\text{pertinent}(e)$. We have that $lcc(e) = \delta$, $lsc(e) = \beta$, and $hsc(e) = \alpha$. If the edge $e'$ is removed from $\text{pertinent}(e)$ then $lsc(e)$ becomes $\delta$. 
Chapter 4

A Linear Time Test for C-Connected Graphs

This chapter proposes a linear time $c$-planarity testing and embedding algorithm for $c$-connected clustered graphs. [CDF+06b] The algorithm is based on the characterization proposed in Chapter 3.

4.1 Testing and Embedding Algorithm: Biconnected Case

In this section we describe a linear time algorithm for testing the $c$-planarity and computing a $c$-planar embedding for $c$-connected clustered graphs whose underlying graph is biconnected. First, we show in Section 4.1 how the $c$-planarity characterizations given in Sections 3.1 and 3.2 can be modified in such a way to produce conditions that are easy to check in linear time. Then, we provide an overview of the algorithm in Section 4.1. Sections 4.1 and 4.1 contain the description of the two main phases of the algorithm.

Encoding the Cluster Hierarchy

The characterizations provided by Theorems 8, 9 and 11 only require to test if a cluster is an ancestor or proper ancestor of another cluster. In fact, we only need to perform comparisons between clusters that are comparable, i.e., that lie on the same path from the root to a leaf of $T$.

Let $\psi$ be a function associating each node $\mu$ of $T$ to a value $\psi(\mu)$ such that $\psi(\mu) > \psi(\nu)$, if $\nu$ is the parent of $\mu$. We can recast the $c$-planarity conditions by replacing
each condition on $T$ with comparisons between suitable values of $\psi$. In the following we adopt as function $\psi(\mu)$ the depth, denoted $d(\mu)$ where the depth of the root of $T$ is zero and $d(\mu) = d(\nu) + 1$ if $\nu$ is the parent of $\mu$.

Observe that the use of the depth instead of the allocation cluster allows to replace several definitions given on the tree $T$ with depth values. Namely, the lowest connecting cluster $lcc(e)$ of a virtual edge $e$ can be replaced by its depth. We denote the value of $d(lcc(e))$ by $d(e)$. Analogously, the lowest connecting cluster $lcc(f)$ of a face $f$ can be replaced by its depth $d(lcc(f))$, denoted $d(f)$. In a similar way we define the highest (lowest) side depth of a virtual edge $e$ as $hsc(e) = d(hsc(e))$ and $lsc(e) = d(lsc(e))$.

According to the above definitions, both the incompatibility of two edges and the conditions of Theorems 9 and 11, can be restated by replacing each occurrence of $\prec$ and $\preceq$ with $<$ and $\leq$, respectively, and by replacing each occurrence of $ac(\cdot)$, $lcc(\cdot)$, $hsc(\cdot)$, and $lsc(\cdot)$ with $d(\cdot)$, $d(\cdot)$, $hsc(\cdot)$, and $lsc(\cdot)$, respectively.

Overview of the Algorithm

The input of the algorithm is a c-connected clustered graph $C(G,T)$ such that $G$ is biconnected and planar. The output of the algorithm is a c-planar embedding of $C$ or a node of the SPQR-tree $T$ for which the c-planarity conditions are not verified. The algorithm consists of two phases that are sketched below and fully described in the following sections.

Preprocessing. This phase consists of three steps.

SPQR-tree Decomposition. First, we compute the depth of each edge $e$ of $G$. Second, we compute an SPQR-tree $T$ of $G$ rooted at any edge $e_r$ of depth zero.

Skeleton-Labelling. We label each non-virtual edge $e$ of the skeletons of $T$ with the three labels $d(e) = hsc(e) = lsc(e)$, which are equal to the depth of the corresponding edge of $G$. Each virtual edge $e$ is labeled with $d(e)$ and $hsc(e)$ only, by performing a suitable bottom-up traversal of $T$.

Edges-Sorting. We sort the edges of each $P$ node of $T$ with respect to the value of their depth and, secondarily, with respect to their highest side depth. The rationale for this sort will be clear later.

Embedding-Construction. We perform a bottom-up traversal of $T$. We check if a non-planarity condition is verified for the current node $\mu$, and in this case we return $\mu$, which is a node of $T$, such that the pertinent graphs of its children are
c-planar but \( \text{pertinent}(\mu) \) is not. Otherwise, we compute a c-planar embedding of skeleton(\( \mu \)), and compute the value \( \text{lsd}(e) \) for the virtual edge \( e \) which represents \( \mu \) in the skeleton of the parent \( \mu' \) of \( \mu \). Finally, we construct the c-planar embedding of the whole graph by means of a top-down traversal of \( T' \).

**The Preprocessing Phase**

The depth of each edge is computed in constant time with a lowest common ancestor query performed with the data structure in [SV88]. The SPQR-tree Decomposition step can be performed in linear time [GM01].

In the Skeleton-Labelling step, we perform a bottom-up traversal of \( T \). Let \( \mu \) be the current node. Based on the values of \( d(e) \) and \( hsd(e) \) of the edges of skeleton(\( \mu \)), we compute the values of \( d(e') \) and \( hsd(e') \) for the virtual edge \( e' \) which represents \( \mu \) in the skeleton of its parent \( \mu' \). The value of \( hsd(e') \) is the minimum of the highest side depth of the edges of \( \mu \). It is easy to see that if \( \mu \) is an S-node (P-node), \( d(e') \) is the minimum (maximum) of the depths of the edges of \( \mu \). If \( \mu \) is an R-node, the computation of \( d(e') \) requires a more detailed analysis of skeleton(\( \mu \)).

**Lemma 11** Let \( \mu \) be an R-node and let MST be a maximum spanning tree of skeleton(\( \mu \)), where the edges are weighted with their depth. The depth of the path with maximum depth between the poles of \( \mu \) is the minimum depth of the edges in the unique path \( p \) in MST between the poles of skeleton(\( \mu \)).

**Proof.** By definition the depth \( d(p) \) is equal to \( d(lcc(p)) \), i.e., the minimum depth of its edges. Let \( e \) be an edge of \( p \) with depth \( d(e) = d(p) \). Suppose, for contradiction, that there is a second path \( p' \) with \( d(p') > d(e) = d(p) \). All edges in \( p' \) have depth greater of \( d(e) \). When \( e \) is removed, MST splits into two trees \( T_u \) and \( T_v \), one containing the pole \( u \) and the other containing the pole \( v \). Each vertex of skeleton(\( \mu \)) either falls into \( T_u \) or into \( T_v \). Since \( p' \) connects \( u \) with \( v \) it necessarily contains an edge \( e' \) which joins a vertex in \( T_u \) with a vertex in \( T_v \). If \( e' \) is chosen to replace \( e \), \( T_u \) and \( T_v \) are joined into tree \( T \), which has weight greater than MST, contradicting the hypothesis that MST is the maximum spanning tree. \( \square \)

Since skeleton(\( \mu \)) is planar and weighted with integer values, a maximum spanning tree can be constructed in linear time (see for example [CT74, Mat95]) with respect to the size of skeleton(\( \mu \)). Hence, because of Lemma 11 the whole Skeleton-Labelling step can be performed in linear time.

The Edges-Sorting step requires special care. In fact, if we performed a separate bucket sort for each P node, since there are instances where the depth has \( O(n) \) values, where \( n \) is the number of vertices of \( G \), in the worst case we spent quadratic time.
Hence, we do the following. First, we construct a unique set $E_P$ of the virtual edges of all the $P$ nodes, each $e$ labelled with $d(e)$, $hsd(e)$, and with its $P$ node. Second, we perform a bucket sort of $E_P$ with respect to $hsd(e)$. Third, we perform a second bucket sort with respect to $d(e)$ considering the virtual edges in the order obtained by the first sort. At this point we have that the elements of $E_P$ are sorted according to the value of their depth and, secondarily, with respect to their highest side depth. Finally, we scan $E_P$ and distribute the edges in their proper skeletons. All this requires linear time.

**The Embedding-Construction Phase**

In the **Embedding-Construction** phase we first perform a bottom-up traversal of $T$ in which the c-planarity conditions are verified for each node $\mu$ and $T$ is decorated with suitable embedding descriptors. Secondly, we perform a top-down traversal of $T$ producing a c-planar embedding for graph $G$ taking into account the values computed for each node $\mu$ of $T$.

Let $\mu$ be the current node in the bottom-up traversal of $T$, let $u$ and $v$ be its poles (assumed arbitrarily ordered at the beginning of the computation), and let $e'$ be the virtual edge which represents $\mu$ in the skeleton of its parent $\mu'$ and let $high(e')$ be a label associated to it. Suppose $skeleton(\mu)$ has been embedded and let $\Gamma_{\mu}$ be its c-planar embedding. We denote right (left) the side that remains on the right (left) hand when traversing clockwise (counterclockwise) the external face of $\Gamma_{\mu}$ from $v$ to $u$. When computing $\Gamma_{\mu}$ we assign to $high(e')$ a value in \{right, left\} which denotes which one between the right and left sides of $\Gamma_{\mu}$ corresponds in pertinent($\mu$) to a path containing an edge $e$ with $d(e) = hsd(e')$. Hence, when processing node $\mu'$, we use $high(e')$ to compute the Boolean value of $flip(\mu)$, that specifies if $\Gamma_{\mu}$ has to be reversed when inserted into $\Gamma_{\mu'}$ in the final top-down traversal.

Provided that the conditions stated in Theorem 9 hold for node $\mu$, we compute an embedding $\Gamma_\mu$ of $skeleton(\mu)$ (if more than one embedding is possible) and the values $flip(\mu_1), \ldots, flip(\mu_k)$ for its children nodes $\mu_1, \ldots, \mu_k$, in such a way to minimize $lsd(e')$. In the following it is specified how $S$, $P$ and $R$ nodes are processed.

**Embedding Construction for S Nodes.**

If $\mu$ is an S-node skeleton($\mu$) has a fixed embedding. We set $flip(\mu_1), \ldots, flip(\mu_k)$ so that the corresponding $high(e_1), \ldots, high(e_k)$ are turned towards the same side of $\Gamma_\mu$, say right. Consequently, the left side has minimum depth $lsd(e') = \min_i lsd(e_i)$. 

42
Embedding Construction for $R$ Nodes.

Suppose $\mu$ is an $R$ node, with children $\mu_1, \ldots, \mu_k$. Let $\Gamma_\mu$ be the (unique) embedding of $\text{skeleton}(\mu)$.

We have to test the c-planarity of $\Gamma_\mu$, and to verify that for each edge $e$ of $\text{skeleton}(\mu)$ incident to two faces $f_1$ and $f_2$ of $\Gamma_\mu$, with $d(f_1) \leq d(f_2)$, if $d(f_1) \leq hsd(e)$ and $d(f_2) \leq lsd(e)$ (see Theorem 9).

Consider the plane graph $G^*$ obtained from $\Gamma_\mu$ by splitting each edge $e$ of $\Gamma_\mu$ with a vertex of depth $d(e)$. It is easy to see that the embedding of $\text{skeleton}(\mu)$ is c-planar if and only if $G^*$ is c-planar.

In order to test the c-planarity of a c-connected clustered graph $C(G, T)$, where $G$ has a fixed embedding $\Gamma$, we rely on Theorem 8. The statement of Theorem 8 requires to check every cycle of $G$ in order to prove the c-planarity of $\Gamma$. This, of course, is not efficient, since we have an exponential number of cycles in a plane graph. Observe, however, that the possible values of $ac(c)$ are as many as the nodes of $T$. Hence, Theorem 8 can be reformulated as follows:

**Lemma 12** An embedding $\Gamma$ of a c-connected clustered graph $C(G, T)$ is c-planar if and only if there is no node $\alpha$ of $T$ such that $G(\alpha)$, induced by the vertices in $\alpha$, contains a cycle $c$ that encloses an edge that is not in $G(\alpha)$.

Let $C(G, T)$ be a c-connected clustered graph where $G(V, E)$ is embedded, let $d_{\text{max}}$ be the height of $T$, and let $D(V', E')$ be the dual graph of $G$. For each $e \in E'$, weight $e$ with the depth of the corresponding primal edge. For each integer $i \in [0, d_{\text{max}}]$, we define the $i$-restricted dual $D_i$ as the subgraph of $D$ containing only edges with weight at most $i$ and no isolated vertex.

**Theorem 12** Let $C(G, T)$ be a c-connected clustered graph and let $d_{\text{max}}$ be the height of $T$. An embedding $\Gamma$ of $G$ is c-planar if and only if:

1. for each integer $i \in [0, d_{\text{max}}]$, graph $D_i$ is connected and
2. an edge $e_r$ of the root of $T$ is on the external face.

**Proof.** First, we prove the necessity of Conditions 1 and 2. Suppose that no edge of the root of $T$ is on the external face of $\Gamma$. By Property 4 there is at least one edge $e_r$ of the root of $T$ in $G$. Hence, the lowest common cluster of the edges on the external face includes edge $e_r$, and Theorem 8 applies. Suppose that the graph $D_k$ is not connected for a depth $k$ in $[0, d_{\text{max}}]$. Since by definition $D_k$ has no isolated vertex, each connected component of $D_k$ contains at least one edge. Denote with $C_r$ the connected component containing an edge $e_r$ on the external face and denote with $e'_r$ an edge contained into
a connected component $C' \neq C_r$. Consider all edges of $D$ attached to a vertex of $C'$ which are not in $C'$. These edges are not in $D_k$ and the corresponding edges of $G$ form a cycle $c$. By Property 3, we have that edges in $c$ can not be shared between two clusters of level $k$. Hence, there exists a cluster $\alpha$ of level $k$ containing the cycle $c$ which separates edges $e_r$ and $e'$, not belonging to $T_\alpha$. Since $e_r$ is on the external face, $e'$ is enclosed by $c$ and Lemma 12 applies.

On the contrary, suppose that the embedding $\Gamma$ is not c-planar. We show that both Conditions 1 and 2 can not be verified. By Lemma 12 there exists a node $\alpha$ of $T$ such that the subtree $T_\alpha$ contains a cycle $c$ that encloses an edge $e$ which is not in $T_\alpha$. Consider a path $p$ connecting $e$ to $c$. By Property 3, $p$ has an edge $e'$, enclosed in $c$, that belongs to a proper ancestor of $\alpha$. By Condition 2 and by the fact that $e_r$ is not part of $c$, we have that $e_r$ is not enclosed by $c$. Hence, each path of $D$ connecting the two edges corresponding to $e_r$ and $e'$ uses at least one edge corresponding to an edge of $c$. It follows that $D_k$ is not connected.

A result similar to Theorem 12 has been presented in [Dah98]. We have the following lemma.

**Lemma 13** Let $G$ be an embedded planar graph, let $D$ be its dual with edges weighted with the depths of the corresponding edges of $G$. Each $i$-restricted dual $D_i$, with $i \in [0, d_{\text{max}}]$, is connected if and only if the minimum spanning tree $\text{mST}(D)$ of $D$, rooted at any vertex $v_r$ of $D_0$, is such that edges of non-decreasing weights are encountered when traversing each path $p$ from $v_r$ to a leaf.

**Proof.** First observe that the $i$-restricted duals $D_i$, for $i \in [0, d_{\text{max}}]$, are the subgraphs of $D$ restricted to the faces and the edges with weight less or equal than $i$, where each face is given the minimum weight of its incident edges. Also, observe that a weighted graph $H$ is connected if and only if it admits a (minimum) spanning forest $\text{mSF}(H)$ which is a single (minimum) spanning tree $\text{mST}(H)$. Therefore, in order to check if each $D_i$ is connected we could test whether it admits a minimum spanning tree $\text{mST}(D_i)$. Further, since we weighted the edges of $D$ with the depth of the corresponding edges of $G$, we have that $\text{mSF}(D_i)$ is a subgraph of $\text{mST}(D_{i+1})$.

If $\text{mSF}(D_i)$ is not connected for some $i$ then each path in $D_k$ connecting two nodes on two different components of $\text{mSF}(D_i)$ uses at least one edge of weight greater than $i$. Hence, all paths connecting $v_r$ to a node $v$ that belongs to a different component (tree) of $\text{mSF}(D_i)$ have at least one edge with weight greater than $i$. It follows that the minimum weight path between $v_r$ and $v$ is not monotonically non-decreasing. Suppose now that $\text{mST}(D_k)$ has a path $p$ from $v_r$ to a leaf which is not monotonically non-decreasing, i.e., $p$ contains at least a sequence of edges of weight $j$ preceded by edge $e_1$ with weight $w_1 < j$ and followed by edge $e_2$ with weight $w_2 < j$. Let $i$ be the
maximum between \( w_1 \) and \( w_2 \). Since \( mSF(D_i) \) is a subgraph of \( mST(D_k) \), we have that \( mSF(D_i) \) contains \( e_1 \) and \( e_2 \), but does not contain the path \( p \), hence it is not connected.

The conditions of Lemma 13 can be used to check the c-planarity of the embedding of the plane graph \( G^* \) in linear time. Let \( D^* \) be the dual of \( G^* \). We compute a minimum spanning tree \( mST(D^*) \) of \( D^* \). As \( D^* \) is planar, \( mST(D^*) \) can be constructed in \( O(n^*) \), where \( n^* \) is the number of nodes of \( D^* \) [CT74, Mat95]. Then, we easily check in \( O(n^*) \) time that the depths are monotonically non-increasing when traversing \( mST(D^*) \) from the root to the leaves.

Consider each children \( \mu_i \) corresponding to \( e_i \). Edge \( e_i \) is incident to two faces, \( f_1 \) and \( f_2 \) for which we assume w.l.o.g. \( d(f_1) \leq d(f_2) \). If \( d(f_1) > hsd(e) \) or \( d(f_2) > lsd(e) \) the algorithm fails since the graph is not c-planar. The value of \( high(\mu_i) \) identifies one of the two faces of \( e_i \), we call it \( f_{high} \). We distinguish two cases: (i) \( f_{high} \) is an internal face of \( \Gamma_\mu \). If \( f_1 = f_{high} \) then we set \( flip(\mu_i) = false \), otherwise \( flip(\mu_i) = true \). (ii) \( f_{high} \) is the external face. We preferentially embed the lowest side into an internal face. Namely, let \( f_{low} \) be the opposite face of \( f_{high} \) with respect of \( e_i \). If \( d(f_{low}) \leq hsd(e) \) then \( flip(\mu_i) = true \) otherwise \( flip(\mu_i) = false \). This can be done in linear time.

We compute \( lsd(e') \) and \( high(e') \) in the following way. We consider the ordered split pair \( \{u,v\} \) of \( e' \) and we call \( b_i \ (b_i) \) the path on the external face of \( \Gamma_\mu \) connecting \( u \) to \( v \) clockwise (counterclockwise). For each edge \( e_i \) on \( b_i \), let \( w_{r,i} \ (w_{l,i}) \) be the depth of the side of \( e_i \) to be turned towards the external face according to \( flip(e_i) \) computed above and \( d_r = min_i w_{r,i} \ (d_l = min_i w_{l,i}) \). If \( d_l < d_r \), we set \( lsd(e') = d_r \) and \( high(e') = left \) otherwise we set \( lsd(e') = d_l \) and \( high(e') = right \). Observe that, the procedure according to which \( flip(\mu_i) \) are computed assures that the embedding described is one with maximum value of \( lsd(e') \) among the possible c-planar embeddings of \( pertinent(e') \).

**Embedding Construction for P Nodes**

If \( \mu \) is a \( P \) node, we have to test the conditions stated in Theorem 9 for \( P \) nodes. If all the conditions hold, we construct a c-planar embedding for \( skeleton(\mu) \) which maximizes the value of \( lsd(e') \), otherwise the graph is not c-planar. Thanks to the **Preprocessing** phase, we have a list \( I(\mu) \) where all the virtual edges of \( skeleton(\mu) \) appear ordered with respect to the \( \preceq_e \) relationship defined as follows: an edge \( e_1 \) precedes \( e_2 \) \( (e_1 \preceq_e e_2) \) if \( d(e_1) > d(e_2) \) or if \( d(e_1) = d(e_2) \) and \( hsd(e_1) \geq hsd(e_2) \).

Condition (a) of Theorem 9 asks to check that \( skeleton(\mu) \) does not contain three pairwise incompatible edges. This can be done by considering the graph of the incom-
4. A Linear Time Test for C-Connected Graphs

patibilities between edges and checking whether this graph is bipartite. Let \( e_1 \) be the first element of \( I(\mu) \). Condition (b) of Theorem 9 asks to test for each edge \( e \in I(\mu) \), with \( e \neq e_1 \), if \( d(e) = lsd(e) \). Also, Condition (b) asks to test for each edge \( e \in I(\mu) \), with \( e \neq e_1 \), if \( d(e) \leq lsd(e_1) \). All these tests can be easily done in time linear in the size of skeleton(\( \mu \)).

The construction of the embedding of skeleton(\( \mu \)) consists of the computation of the order of the vertices of \( \mu \). Namely, the proof of Theorem 9 ensures that a c-planar embedding of skeleton(\( \mu \)) is such that edges are ordered into two sequences \( I_L = \langle e_1 \succeq e_2 \succeq e_3 \cdots \succeq e_p \rangle \) and \( I_R = \langle e_r \succeq e_r \succeq e_r \cdots \succeq e_r \rangle \), each one composed by compatible edges. The fact that the incompatibility graph is bipartite ensures the existence of \( I_L \) and \( I_R \). Further, since we want to maximize the value of \( lsd(e') \), we search for a particular pair \( I_L \) and \( I_R \) such that the difference between \( \max_{e \in I_L} hsd(e) \) and \( \max_{e \in I_R} hsd(e) \) is maximized.

The computation of \( I_L \) and \( I_R \) requires the use of the following lemma.

**Lemma 14** Let \( I \) be a sequence of virtual edges ordered with respect to the \( \preceq_e \) relationship, such that edges in \( I \) are pairwise compatible. Suppose \( e \notin I \) is an edge following all edges in \( I \) with respect to the \( \preceq_e \) relationship. If \( e \) is compatible with the last edge in \( I \) then \( e \) is compatible with all edges in \( I \).

**Proof.** Let \( e_{\text{last}} \) be the last edge in \( I \). Since \( e \) is compatible with \( e_{\text{last}} \) and \( e_{\text{last}} \preceq_e e \), we have that \( d(e) \leq hsd(e_{\text{last}}) \). Since all the edges in \( I \) are pairwise compatible, we also have that \( d(e_{\text{last}}) \) is less or equal than the highest side depth of all edges in \( I \). It follows that \( d(e) \) is less or equal than the highest side depth of each edge in \( I \), and therefore \( e \) is compatible with all edges in \( I \). \( \square \)

We build two sequences \( I_1 \) and \( I_2 \) by inserting one by one the edges of \( I(\mu) \) into one of them. Namely, we start by inserting \( e_1 \) in \( I_1 \). Let \( e_i \) be the current edge and let \( e_{1,\text{last}} \) and \( e_{2,\text{last}} \) be the last inserted elements of \( I_1 \) and \( I_2 \), respectively. If \( e_i \) is incompatible with the last element of one of the two sequences we insert it into the other sequence. Otherwise, if \( e_i \) is compatible with both \( e_{1,\text{last}} \) and \( e_{2,\text{last}} \), then we insert it into the sequence containing \( \min\{hsd(e_{1,\text{last}}), hsd(e_{2,\text{last}})\} \). We set \( I_L \) as the reverse of \( I_1 \) and \( I_R = I_2 \).

Since we insert an edge \( e_i \) into a sequence only if \( e_i \) is compatible with the last element of the sequence, and the sequences are ordered with respect to the \( \preceq_e \) relationship, Lemma 14 ensures that both \( I_L \) and \( I_R \) contain pairwise compatible edges. If an edge \( e \) is compatible with both the sequences, inserting it into the sequence with smaller value of highest side depth on the last edge guarantees that the difference between \( \max_{e \in I_L} hsd(e) \) and \( \max_{e \in I_R} hsd(e) \) is maximized. In fact, the following property holds:
Property 9  Let \( I \) be a sequence of edges ordered with respect to the \( \preceq_e \) relationship, such that edges in \( I \) are pairwise compatible. The last edge \( e_{\text{last}} \) in \( I \) has \( hsd(e_{\text{last}}) = \max_{e \in I}(hsd(e)) \).

According to the construction rules provided in the sufficiency proof of the characterization given in Subsection 3.1, for each edge \( e_i \in I_L \), we set \( \text{flip}(e_i) = \text{true} \) if \( \text{high}(e_i) = \text{right} \), and \( \text{flip}(e_i) = \text{false} \) otherwise. Conversely, for each edge \( e_i \in I_R \), we set \( \text{flip}(e_i) = \text{true} \) if \( \text{high}(e_i) = \text{left} \), and \( \text{flip}(e_i) = \text{false} \) otherwise. Finally, the value of \( lsd(e') \) is maximum between \( hsd(e_l) \) and \( lsd(e_r) \). All the operations performed on a \( P \) node can be clearly executed in linear time.

Finally, we compute the c-planar embedding of \( G \). We start with the current embedding equal to the skeleton of the child of the root of \( T \) and proceed by means of a top-down traversal of \( T \). For each node \( \mu \) of \( T \) with children \( \mu_1, \ldots, \mu_k \), the embeddings of \( \text{skeletons}(\mu_i) \) are merged into the current embedding. If \( \text{flip}(\mu_i) = \text{true} \) the embedding is flipped before the merge operation. This computation is linear since each skeleton is flipped at most once.

The whole algorithm is summarized in Figures 4.2, 4.3, 4.4, 4.5, and 4.6.

From the above discussion we can state the following theorem.

Theorem 13  Given a c-connected clustered graph \( C(G,T) \), such that \( G \) is biconnected, the above described algorithm tests the c-planarity of \( C \), and, if \( C \) is c-planar, computes a c-planar embedding of \( C \) in linear time.

4.2  Testing and Embedding Algorithm: General Case

In this section we extend the algorithm presented in Section 4.1 to the case of c-connected clustered graph whose underlying graph is planar and simply connected.

The following lemmas permit to state the correctness of the algorithm.

Lemma 15  Let \( C(G,T) \) be a c-planar clustered graph and let \( B \) be the block-cutvertex tree of \( G \). Let \( \alpha \) be a cutvertex of \( B \) with parent \( \mu \) and let \( \{u, \alpha\} \) be a split pair of \( \mu \). Suppose that in a c-planar embedding of \( C \) pertinent(\( \alpha \)) appears in an internal face of the embedding of pertinent(\( u, \alpha \)). There exists a c-planar embedding of \( C \) such that pertinent(\( \alpha \)) is embedded in the external face of the embedding of pertinent(\( u, \alpha \)).

Proof. Suppose that there is no c-planar embedding of \( G \) unless pertinent(\( \alpha \)) is inside pertinent(\( u, \alpha \)). This implies that in any drawing of \( C \) with pertinent(\( \alpha \)) embedded outside pertinent(\( u, \alpha \)) at least one of the following two conditions is verified:
4. A Linear Time Test for C-Connected Graphs

(i) there is a cycle \( c \) of depth \( d(c) > d(\text{pertinent}(u, \alpha)) \) enclosing \( \text{pertinent}(u, \alpha) \); (ii) there are two cycles \( c_1 \) and \( c_2 \) of depth greater than \( d(\text{pertinent}(\alpha)) \) passing through \( \text{pertinent}(u, \alpha) \) and enclosing the two faces outside \( \text{pertinent}(u, \alpha) \) (see the dotted and dashed cycles of Fig. 4.1). In case (i), since \( c \) encloses both the faces outside \( \text{pertinent}(u, \alpha) \), there can not be a c-planar embedding with \( \text{pertinent}(\alpha) \) inside \( \text{pertinent}(u, \alpha) \). In case (ii), from Fig. 4.1 it is apparent that the parts of the two cycles \( c_1 \) and \( c_2 \) outside \( \text{pertinent}(u, \alpha) \) form a cycle enclosing \( \text{pertinent}(u, \alpha) \). Hence, there can not be a c-planar embedding with \( \text{pertinent}(\alpha) \) inside \( \text{pertinent}(u, \alpha) \).

\[ \square \]

**Lemma 16** Let \( C(G, T) \) be a c-planar clustered graph and let \( B \) be the block-cutvertex tree of \( G \). Let \( \alpha \) be a cutvertex of \( B \) with children \( \mu_1 \) and \( \mu_2 \). Suppose that in a c-planar embedding of \( C \) \( \text{pertinent}(\mu_2) \) appears in an internal face of the embedding of \( \text{pertinent}(\mu_1) \). There exists a c-planar embedding of \( C \) such that \( \text{pertinent}(\mu_2) \) appears in the external face of the embedding of \( \text{pertinent}(\mu_1) \).

**Proof.** Suppose that there is no c-planar embedding of \( G \) unless \( \text{pertinent}(\mu_2) \) is not placed inside a face of \( \text{pertinent}(\mu_1) \). This implies that in any drawing of \( C \) with \( \text{pertinent}(\mu_2) \) embedded outside \( \text{pertinent}(\mu_1) \) there is a cycle \( c \) of depth \( d(c) > d(\text{pertinent}(\mu_2)) \) enclosing \( \text{pertinent}(\mu_2) \). Since \( c \) necessarily encloses \( \mu_1 \) and \( \mu_2 \),
there can not be a c-planar embedding of $C$ such that $pertinent(\mu_2)$ is placed inside a face of $pertinent(\mu_1)$. \qed

We now show a linear-time algorithm for testing and embedding a general c-connected clustered graph.

**BC-tree Decomposition.** First, for each edge $e$ of $G$ we compute $d(e)$. Second, we compute the BC-tree $\mathcal{B}$ of $G$ and root $\mathcal{B}$ to a block $v$ containing an edge $\tau$ such that $d(\tau) = 0$.

**BC-tree Labelling.** We traverse $\mathcal{B}$ bottom-up and compute for each cutvertex $\rho_i$ the depth of $pertinent(\rho_i)$. This is done by taking the minimum depth of the pertinent of the children blocks of $\rho_i$.

**Block Preprocessing** We perform a second bottom-up traversal of $\mathcal{B}$ and execute on each block $\mu$ a variation of the Preprocessing phase for biconnected graphs, where the sorting phase is factored out and cut-vertices are considered. Namely, for each block $\mu$ the following two steps are performed.

**SPQR-tree Decomposition.** First, we compute an SPQR-tree $\mathcal{T}_\mu$ rooted at any edge $e$, whose depth is the minimum depth of the block.

**Skeleton Labelling.** For each node $\sigma$ in $\mathcal{T}_\mu$, consider each edge $e$ of $skeleton(\sigma)$ such that $pertinent(e)$ is a single edge $e'$. We label $e$ such that $hsd(e) = lsd(e) = d(e) = d(e')$. We perform a bottom-up traversal of $\mathcal{T}_\mu$ in order to label each virtual edge $e$ with $d(e)$ and $hsd(e)$. Let $e$ be a virtual edge of any skeleton. The value of $d(e)$ is computed with the same operations used for biconnected graphs. Let $\rho_1, \ldots, \rho_k$ be the cutvertices of $\mu$ contained in $skeleton(e)$ that are not poles of $e$, possibly comprehensive of the parent of $\mu$. The value of $hsd(e)$ is the minimum of the highest side depths of the edges of $skeleton(e)$ and the depths of $pertinent(\rho_i)$. This implies that the parent cutvertex of $\mu$ is adjacent to a face $f$ with lowest depth in the computed embedding for $\mu$. As stated in 11 the external face can be changed so that the parent cutvertex is incident to the external face and hence the condition of Theorem 11, modified as in Section 4.1, is verified.

**Edges Sorting.** We simultaneously sort the edges of all $P$ nodes of all the computed SPQR-trees with respect to the value of their depth, and secondarily with respect to their highest side depths. We use a strategy analogous to that used for biconnected graphs in order to preserve the linearity of this algorithmic step.
4. A Linear Time Test for C-Connected Graphs

**Block Embedding Construction.** For each block $\mu$ we consider its SPQR-tree $T_\mu$ and perform a bottom-up traversal of it. We check if a non-planarity condition (see Theorem 9) is verified for the current node $\sigma$, possibly computing a c-planar embedding of $\text{skeleton}(\sigma)$ and the value of $\text{lsd}(e)$ for the virtual edge $e$ which represents $\sigma$ in the skeleton of its parent $\sigma'$. In the case $\sigma$ is a $P$ node, the test of the c-planarity conditions, the computation of the embedding of $\text{skeleton}(\sigma)$, and the computation of $\text{lsd}(e)$ follow the same rules described for biconnected graphs (see Section 4.1).

In the case $\sigma$ is an $S$ node, we proceed as for biconnected graphs. Plus, consider each vertex $\rho$ of $\text{skeleton}(\sigma)$ which is also a cutvertex and is not a pole of $\sigma$. All the blocks that are children of $\rho$ in $B$ are embedded in the side where all the highest sides of the children of $\sigma$ in $T$ are embedded. The correctness of this approach is implied by Lemmas 15 and 16.

In the case $\sigma$ is an $R$ node, the existence of cutvertices in $\text{skeleton}(\sigma)$ must be taken into account. Besides the tests performed for the biconnected case we have to make sure that the second condition of Theorem 11, modified as in Section 4.1, is verified. Namely, each cutvertex $\rho$ that is not a pole of $\sigma$ must be incident to a face $f$ of $\text{skeleton}(\sigma)$ with $d(f)$ less or equal than the depth of $\text{pertinent}(\rho)$. When choosing $f$, an internal face is always preferred if it respects this condition. All blocks that are children of $\rho$ in $B$ are embedded in $f$. The correctness of this approach is implied by Lemmas 15 and 16. If such a face does not exist the algorithm fails since the graph is not c-planar.

We compute $\text{flips}(\cdot)$ of the children of $\sigma$ as for biconnected graphs. When computing $\text{lsd}(e')$ and $\text{high}(e')$ we proceed as for the biconnected graphs but for the computation of $d_l$ and $d_r$, see Section 4.1 **Embedding Construction for R Nodes.** Namely, the computation of $d_r$ ($d_l$) must take into account the depth of the cutvertices in $b_r$ ($b_l$) that have their blocks embedded in the external face of $\text{skeleton}(\sigma)$.

Observe that, as in the biconnected case, the adopted procedure assures that the embedding described by $\text{flip}(\cdot)$ and by the choices on the cutvertices, is one with minimum value of $\text{lsd}(e')$ among the possible c-planar embeddings of $\text{pertinent}(e')$.

In the case $\sigma$ is the unique child of the root of $T_\mu$ with poles $u$ and $v$, besides the regular operations described above, we check if $u$ or $v$ are cutvertices and embed all their blocks in the external face.

The reporting of the embedding of $\mu$ is performed as for biconnected graphs.
Testing and Embedding Algorithm: General Case

**Block Re-rooting and Merging.** We consider the computed embedding $\Gamma_\mu$ of each block $\mu$ of $B$ and we adopt as external face of $\Gamma_\mu$ a face with minimum depth incident to the parent cutvertex of $\mu$. We merge together the obtained embeddings of the blocks.

The whole algorithm is summarized in Figures 4.7 and 4.8. Due to the above description the following theorem holds.

**Theorem 14** The $c$-planarity of a $c$-connected clustered graph can be tested, and possibly a $c$-planar embedding can be built, in linear time.
C-planarity algorithm for biconnected graphs - Part 1

**input:** A c-connected clustered graph $C(G, T)$, where $G$ is a planar biconnected graph

**output:** A c-planar embedding of $G$ if $C$ is c-planar, a triconnected component causing non-c-planarity otherwise

**Preprocessing Phase**

```plaintext
for all edge $e \in G$ do
    compute $d(e)$, $hsd(e)$, $lsd(e)$
end for

calculate the SPQR tree $T$ of $G$, rooted to an edge with $d(e) = 0$
for all node $\mu$ in $T$ in post-order traversal do
    let $e'$ be the virtual edge representing $\mu$ in the skeleton of its parent node.
    $hsd(e') = \min_{e \in \text{skeleton}(\mu)} hsd(e)$
    if $\mu$ is an $S$ node then
        $d(e') = \min_{e \in \text{skeleton}(\mu)} d(e)$
    else if $\mu$ is a $P$ node then
        $d(e') = \max_{e \in \text{skeleton}(\mu)} d(e)$
    else if $\mu$ is an $R$ node then
        Compute a Maximum Spanning Tree MST of skeleton(\mu)
        Let $p$ be the path between the poles in MST.
        $d(e') = d(p)$
    end if
end for
sort the edges of each $P$ node using a unique bucket sort.
```

Figure 4.2: The c-planarity testing and embedding algorithm for c-connected clustered graphs whose underlying graph is biconnected - Part 1.
C-planarity algorithm for biconnected graphs - Part 2

**input:** A c-connected clustered graph $C(G, T)$, where $G$ is a planar biconnected graph

**output:** A c-planar embedding of $G$ if $C$ is c-planar, a triconnected component causing non-c-planarity otherwise

**Embedding Construction Phase**

for all node $\mu$ in $T$ in post-order traversal do

if $\mu$ is an $S$ node then

    for all $e \in \text{skeleton}(\mu)$ do

        if high($e$) = left then
            flip($e$) = true
        else
            flip($e$) = false
        end if

    end for

    $\text{lsd}(e') = \min_{e \in \text{skeleton}(\mu)} \text{lsd}(e)$
    $\text{high}(e') = \text{right}$

else if $\mu$ is an $P$ node then

    if $\text{ProcessPNode}(\mu, e') = \text{False}$ then
        return $\mu$
    end if

else if $\mu$ is an $R$ node then

    if $\text{ProcessRNode}(\mu, e') = \text{False}$ then
        return $\mu$
    end if

end if

end for

construct the c-planar embedding by performing a top-down traversal of $T$ and considering values of flip(.)

return the embedding of $G$

Figure 4.3: The c-planarity testing and embedding algorithm for c-connected clustered graphs whose underlying graph is biconnected - Part 2.
4. A Linear Time Test for C-Connected Graphs

Procedure ProcessPNode(µ, e′)

{The edges of skeleton(µ) are already ordered in a list I(µ)}
let e₁ be the first element of I(µ)
if skeleton(µ) contains three pairwise incompatible edges then
    return False
end if
for all e ≠ e₁ in skeleton(µ) do
    if d(e) ≠ lsd(e) or d(e) > lsd(e₁) then
        Return False
    end if
end for
initialize lists I_L = {e₁} and I_R = {}
for all e ≠ e₁ in skeleton(µ) do
    e_l = last element in I_L, e_r = last element in I_R
    if e is incompatible with e_l then
        append e to I_R
    else if e is incompatible with e_r then
        append e to I_L
    else
        append e to the list containing min{hsd(e_l), lsd(e_r)}
    end if
end for
the embedding of skeleton(µ) is I_LI_R, where I_L is the reverse of I_L
for all e in I_L do
    if high(e) ≠ left then
        flip(e) = true
    end if
end for
for all e in I_R do
    if high(e) ≠ right then
        flip(e) = false
    end if
end for
lsd(e′) = max{min_e∈I_L hsd(e), min_e∈I_R hsd(e)}
if hsd(e_l) ≤ hsd(e_r) then
    high(µ) = left else high(µ) = right
end if
return True

Figure 4.4: Testing and embedding procedure for P nodes.
Procedure ProcessRNode(µ, e') - Part 1

construct the graph $G^*$ from skeleton(µ)
compute the planar embedding of $G^*$ with the poles on the external face
compute the dual graph $D$ of $G^*$
compute the minimum spanning tree $mST$ of $D$
if $mST$ is non monotonic then
    return False
end if
for all $e$ in skeleton($\mu$) do
    let $f_1$ and $f_2$ be the faces incident to $e$, with $d(f_1) \leq d(f_2)$
    if $hsd(e) < d(f_1)$ or $lsd(e) < d(f_2)$ then
        return False
    end if
    Let $f_{high}$ be the face incident to $e$ identified by $high(e)$
    if $f_1$ is the external face AND $hsd(e) \geq d(f_2)$ then
        if $f_1 = f_{high}$ then
            $flip(e) = true$ else $flip(e) = false$
        end if
    else
        if $f_1 \neq f_{high}$ then
            $flip(e) = true$ else $flip(e) = false$
        end if
    end if
end if
end for

Figure 4.5: Testing and embedding procedure for $R$ nodes - Part 1.
4. A Linear Time Test for C-Connected Graphs

Procedure ProcessRNode\( (\mu, e') \) - Part 2

\[
\begin{align*}
\text{let } \{u, v\} \text{ the ordered split pair of } e' \\
\text{let } b_r \text{ the path on the external face of skeleton}(\mu) \text{ connecting } u \text{ to } v \text{ clockwise} \\
\text{let } b_l \text{ the path on the external face of skeleton}(\mu) \text{ connecting } u \text{ to } v \text{ counterclockwise.} \\
\text{for all } e_i \in b_r \text{ do} \\
\quad \text{let } w_{r,i} \text{ be the depth of the side of } e_i \text{ to be turned towards the external face} \\
\text{end for} \\
\text{for all } e_i \in b_l \text{ do} \\
\quad \text{let } w_{l,i} \text{ be the depth of the side of } e_i \text{ to be turned towards the external face} \\
\text{end for} \\
\text{let } d_r = \min_i w_{r,i} \\
\text{let } d_l = \min_i w_{l,i} \\
\text{if } d_l < d_r \text{ then} \\
\quad \text{let } lsd(e') = d_r \\
\quad \text{let } high(e') = left \\
\text{else} \\
\quad \text{let } lsd(e') = d_l \\
\quad \text{let } high(e') = right \\
\text{end if} \\
\text{return true}
\end{align*}
\]

Figure 4.6: Testing and embedding procedure for R nodes - Part 2.
C-planarity testing and embedding algorithm for connected graphs - Part 1

**input:** A c-connected clustered graph $C(G, T)$, where $G$ is a planar graph

**output:** “True” and a c-planar embedding of $G$ if $C$ is c-planar, “False” otherwise

**Block Preprocessing Phase**

**for all edge** $e \in G$ **do**

Compute $d(e), hsd(e), lsd(e)$

**end for**

compute the BC tree $B$ of $G$, rooted to a block containing an edge $e$ with $d(e) = 0$

**for all** cutvertex $\rho$ in $B$ in post-order traversal **do**

compute the depth of $pertinent(\rho)$

**end for**

**for all** node $\mu$ in $B$ in post-order traversal **do**

compute the SPQR tree $T_\mu$ rooted to an edge with minimum depth

For each non virtual edge $e \in T_\mu$ compute $d(e), hsd(e), lsd(e)$

**for all** node $\sigma \in T_\mu$ in post-order traversal **do**

compute $d(\sigma)$ as in the biconnected case

let $\rho_i$ be the cutvertices in $skeleton(\sigma)$ different from the poles

compute $hsd(\sigma) = \min_i \{hsd(e_i), d(pertinent(\rho_i))\}, \text{ with } e_i \in skeleton(\sigma)$

**end for**

**end for**

Sort the edges of each $P$ node of each block with a unique bucket sort

Figure 4.7: The c-planarity testing and embedding algorithm for c-connected clustered graphs - Part 1
C-planarity testing and embedding algorithm for connected graphs - Part 2

**input:** A c-connected clustered graph \( C(G, T) \), where \( G \) is a planar graph

**output:** “True” and a c-planar embedding of \( G \) if \( C \) is c-planar, “False” otherwise

**Block Embedding Phase**

for all node \( \mu \) in \( \mathcal{B} \) do

for all node \( \sigma \in \mathcal{F}_\mu \) in post-order traversal do

let \( \rho_i \) be the cutvertices in \( \text{skeleton}(\sigma) \) different from the poles

if \( \sigma \) is an \( \mathcal{S} \) node then

process \( \sigma \) as in the biconnected case

embed the blocks connected to \( \rho_i \) in the highest side of \( \text{skeleton}(\sigma) \)

else if \( \sigma \) is an \( \mathcal{P} \) node then

process \( \sigma \) as in the biconnected case

else if \( \sigma \) is an \( \mathcal{R} \) node then

test the condition on \( \text{skeleton}(\sigma) \) as in the biconnected case

if each \( \rho_i \) is not incident to a face \( f \) with \( d(f) \leq d(\text{pertinent}(\rho_i)) \) then

return False

else

embed the blocks of \( \rho_i \) in a suitable (possibly internal) face \( f \)

end if

compute the flip for each virtual edge as in the biconnected case

compute \( \text{lsd}(\sigma) \) considering the blocks embedded on the external face

compute \( \text{high}(\sigma) \) considering the blocks embedded on the external face

end if

end for

construct the embedding \( \Gamma_\mu \) of \( \mu \) as in the biconnected case

let \( f \) be a face with minimum depth incident to the root cutvertex of \( \mu \)

choose \( f \) as external face for \( \Gamma_\mu \)

end for

merge the embedding of the blocks

Figure 4.8: The c-planarity testing and embedding algorithm for c-connected clustered graphs - Part 2
Chapter 5

Cycles of Clusters

In this chapter we study simple families of clustered graphs that are highly unconnected. We start by studying 3-cluster cycles, that are flat clustered graphs such that the underlying graph is a simple cycle and there are three clusters. We show that in this case testing the c-planarity can be done efficiently and give an efficient drawing algorithm. Also, we characterize 3-cluster cycles in terms of formal grammars. Finally, we generalize the results on 3-cluster cycles considering clustered graphs that have a cycle structure at each level of the inclusion tree. Even in this case we show efficient c-planarity testing and drawing algorithms.

Figure 5.1: (a) An example of a cycle with labels in \{a, b, c\}. (b) The cycle with extra edges. (c) The corresponding clustered drawing of the cycle.
5. Cycles of Clusters

5.1 Preliminaries

We assume familiarity with formal grammars [HU79].

Given an unconnected clustered graph $C(G,T)$, a saturator of $C$ is a set of edges that can be added to the underlying graph $G$ so that $C$ becomes connected. If $G$ with the added edges is c-planar, we say that the saturator is planar, non-planar otherwise. It is easy to see that an unconnected clustered graph is c-planar iff it admits a planar saturator. Finding a saturator of a clustered graph is important since it allows us to apply to $C$ the same drawing techniques that have been devised for connected clustered graphs.

We define a 3-cluster cycle as a flat clustered graph such that the underlying graph is a simple cycle and there are exactly three clusters (all at the same level), plus the root cluster. In a 3-cluster cycle the inclusion tree consists of a root node with three children. Each vertex of the underlying cycle is a child of one of these three nodes. Given a 3-cluster cycle, we associate a label in $\{a, b, c\}$ with each of the three clusters.

Fig. 5.1.a provides an example of a 3-cluster cycle. The problem of finding a c-planar drawing for the 3-cluster cycle of Fig. 5.1.a is equivalent to the problem of adding new edges so that: (i) the new graph (i.e., cycle plus new edges) is planar and (ii) for each label, the subgraph induced by the vertices with that label is connected. In the case of Fig. 5.1.a the problem admits a solution, depicted in Fig. 5.1.b. The set of edges added to the cycle, which we call the saturator, is used to “simulate” the closed regions containing the clusters (See Fig. 5.1.c). Observe that the clustered graph of Fig. 5.1 is not connected. Observe also that there exist 3-cluster cycles that are not c-planar. Fig. 5.2 provides an example of a non-c-planar 3-cluster cycle.

Consider a 3-cluster cycle and arbitrarily select a starting vertex and a direction. We can visit the cycle and denote it by the sequence $\sigma$ of labels associated with the clusters encountered during the visit. The same 3-cluster cycle is also denoted by any cyclic permutation of $\sigma$ and by any reverse sequence of such permutations. We use Greek letters to denote general sequences and Roman letters to identify single-character sequences. Given a sequence $\sigma$, we denote with $\overline{\sigma}$ its reverse sequence.

It is easy to see that repeated consecutive labels can be collapsed into a single label without affecting the c-planarity property of a 3-cluster cycle. Hence, in the following we consider only 3-cluster cycles where consecutive vertices belong to distinct clusters. Also, since clusters can not be empty, in a 3-cluster cycle at least one occurrence of each label can be found.

We assign a cyclic order to the labels $a, b, c$ so that $a \prec b, b \prec c,$ and $c \prec a$. A sequence $\sigma$ is monotonic increasing (decreasing) if for each pair $x, y$ of consecutive labels of $\sigma$ $x \prec y$ ($y \prec x$). A sequence is cyclically monotonic increasing (decreasing) if all its cyclic permutations are monotonic increasing (decreasing).
Given a 3-cluster cycle $\sigma$, $Balance(\sigma)$ is a number defined as follows. Select a start vertex and a direction. Set counter $C$ to zero. Visit $\sigma$ adding (subtracting) one unit to $C$ when passing from $x$ to $y$, where $x \prec y$ ($y \prec x$). Observe that, when the start vertex is reached again, $C$ is a multiple of 3 that can be positive, negative, or zero. If we selected a different start vertex, while preserving the direction, we would obtain the same value. On the contrary, if $\sigma$ was visited in the opposite direction the opposite value would be obtained for $C$. $Balance(\sigma) = |C|$. For example, $Balance(ababc) = 3$ and $Balance(cbacba) = 6$.

Observe that, when representing a 3-cluster cycle with a sequence of labels, by reading the sequence from left to right, we implicitly choose a direction for visiting the cycle. For simplicity, we adopt the convention of representing a 3-cluster cycle with a sequence $\sigma$ such that, when the vertices of the cycle are visited according to the order induced by $\sigma$, a non-negative value for $C$ is obtained.

### 5.2 Cycles with Three Clusters

In this section we address the problem of testing the c-planarity of a 3-cluster cycle. Lemma 17 introduces a transformation, called “zig-zag removal”, that can be applied to a 3-cluster cycle to obtain a smaller 3-cluster cycle which is c-planar if and only if the starting 3-cluster cycle is c-planar. Lemma 18, by repeatedly applying the above mentioned transformation, shows that any 3-cluster cycle can be reduced to a trivial 3-cluster cycle which is either cyclically monotonic or it is composed by two maximal
monotonic subsequences. Lemma 19 and Lemma 20 show how to test c-planarity for these two kinds of 3-cluster cycles. Finally, Theorems 15, 16, and 17 state the main results about c-planarity of 3-cluster cycles.

**Lemma 17 (Zig-zag removal)** Let \( \sigma = \sigma_1 x \alpha y \alpha_2 x \alpha y \sigma_2 \) be a 3-cluster cycle such that \( \sigma_1, \sigma_2, \) and \( \alpha \) are possibly empty and \( x \alpha y \) is monotonic. The 3-cluster cycle \( \tau = \sigma_1 x \alpha y \sigma_2 \) is c-planar if and only if \( \sigma \) is c-planar. Balance(\( \sigma \)) = Balance(\( \tau \)).

![Figure 5.3: Illustration for the proof of Lemma 17. Circles represent clusters.](image-url)

Before showing a formal proof for this Lemma, we give an intuitive description of the main idea used for the proof.

Suppose there exists a c-planar drawing of \( \tau \). The black line in Fig. 5.3 shows an example of such a drawing for the portion concerning subsequence \( x \alpha y \). Introduce between \( y \) and the first vertex of \( \sigma_2 \) the sequence \( \alpha x \alpha y \), drawing it suitably close to \( x \alpha y \) so to preserve c-planarity. The result is shown in Fig. 5.3 where the added part is drawn gray.

Conversely, suppose that there exists a c-planar drawing of \( \sigma \). Fig. 5.4.a shows an example of such a drawing for the part concerning subsequence \( x \alpha y \alpha_2 x \alpha y \). The inlet formed by \( x \alpha y \alpha_2 \) may contain parts of \( \sigma \) that are denoted by \( Q \) in Fig. 5.4.a. Analogously, the parts of \( \sigma \) that are contained in the inlet formed by \( y \alpha x \alpha y \) are denoted by \( P \). The embedding of \( P \) and \( Q \) may be rearranged preserving c-planarity as shown in Fig. 5.4.b. Path \( \alpha x \alpha y \) can now be replaced with an edge connecting vertex \( y \) with the first vertex of \( \sigma_2 \).

A formal proof for Lemma 17 follows.
Proof. In the following we use primes to distinguish different occurrences of the same character (as in $x'$ and $x''$) or of the same substring (as in $\alpha'$ and $\alpha''$). According to this notation $\sigma = \sigma_1 x' \alpha' y' \alpha'' x'' y'' \sigma_2$ and $\tau = \sigma_1 x' \alpha' y' \sigma_2$.

We denote by $k$ the length of the sequences $\alpha', \tau$, and $\alpha''$. We denote with $\alpha(j)$ the $j$-th vertex of sequence $\alpha$ where $j \in \{1, \ldots, k\}$. Suppose that there exists a $c$-planar drawing $\Gamma_\tau$ of $\tau$. (Fig. 5.5.a shows an example of such a drawing for the part related to the subsequence $x' \alpha' y'$.) We prove that $\sigma$ is $c$-planar by constructing a $c$-planar drawing $\Gamma_\sigma$ in the following way.

All vertices and cluster boundaries of $\Gamma_\tau$ are drawn in $\Gamma_\sigma$ in the same way as in $\Gamma_\tau$. Edges of $\Gamma_\tau$ are drawn in $\Gamma_\sigma$ in the same way as in $\Gamma_\tau$ with the exception of the edge between $y'$ and the vertex $z$ following $y'$ in $\tau$. Such an edge is replaced by the path $\alpha'' x' \alpha' y' \alpha'' x'' y'' \sigma_2$ with $z$ connected to $y'$ and $y'$ connected to $z$. This path is drawn as follows. Vertex $x''$ is placed arbitrarily near to $x'$ in the region of $\Gamma_\sigma$ that is on the left side of the edge joining $x'$ with $\alpha' (1)$. Vertex $y''$ is placed arbitrarily near to $y'$ in the region of $\Gamma_\sigma$ that is on the right side of the edge joining $y'$ with $\alpha' (k)$. For each vertex $\alpha'(j)$ the corresponding vertex $\alpha''(k-j+1)$ is placed arbitrarily near to $\alpha'(j)$ into the region that is on the right side when traversing $\alpha'$ from $x'$ to $y'$. The path $y' \alpha x''$ can now be connected without crossings since the edges of $\alpha'$ can be drawn arbitrarily close to the edges of $\alpha''$ (see Fig. 5.5.b). Analogously, for each vertex of $\alpha''(j)$ the corresponding vertex of $\alpha''(k-j+1)$ is placed arbitrarily near to $\alpha''(j)$ into the region that is on the right side when traversing $\alpha''$ from $y'$ to $x''$. The path $x' \alpha'' y''$ can now be connected without crossings since edges of $\alpha''$ can be drawn arbitrarily close to the edges of $\alpha''$. 
5. CYCLES OF CLUSTERS

(see Fig. 5.5.c). Finally, vertex $y''$ can be connected to $z$, crossing the boundary of cluster $y$ only. The drawing $\Gamma_\sigma$ is c-planar, since the set of edges added to $\Gamma_\tau$ do not generate edge crossings or edge region crossings.

Suppose $\sigma$ is c-planar and $\Gamma_\sigma$ is a c-planar drawing of it. We prove that $\tau$ is c-planar by showing how to build a c-planar drawing $\Gamma_\tau$ starting from $\Gamma_\sigma$.

Call $A_1$ the set of edges of $x'\alpha'y'$, $\overline{A}$ the set of edges of $y'\overline{\alpha}y''$, and $A_2$ the set of edges of $x''\alpha'y''$. In $\Gamma_\sigma$ (see Fig. 5.6.a) the inlet delimited by $y'\overline{\alpha}y''$ and by the part of the border of cluster $y$ between the edges $y'\overline{\alpha}(k)$ and $\alpha''(k)y''$ may contain some parts of $\sigma$. Call $P$ the set of edges of such parts including those that cross the border of the cluster of $y'$. Analogously, consider the inlet delimited by $x'\alpha'y'\overline{\alpha}y''$ and by the part of the border of the cluster of $x'$ between the edges $x'\alpha'(1)$ and $\overline{\alpha}(1)x''$. This inlet

![Diagram](image_url)

Figure 5.5: (a) The drawing of $x'\alpha'y'$ in $\Gamma_\tau$. The gray zones are part of cluster regions. Note that $\tau$ may pass through each cluster many times. (b) After the insertion of path $y'\overline{\alpha}y''$. (c) After the insertion of path $x''\alpha'y''$, the last edge is connected to the first vertex of $\sigma_2$. 
Figure 5.6: (a) The drawing of $x'\alpha'y''\alpha''y''$ in $\Gamma_\sigma$. The gray zones are part of cluster regions. Note that $\tau$ may pass through each cluster many times. (b) The drawing of $\Gamma_\tau$ after $y''\alpha''y''$ was deleted and $P$ was moved

may contain some parts of $\sigma$. Call $Q$ the edges of the subgraph in this inlet including those that cross the boundary of the cluster of $x'$.

To construct $\Gamma_\tau$ we delete the path $\overline{\alpha'}\alpha''y''$. Then, we move the subgraph induced by $P$, which is on the left side of $x'\alpha'y'$, to the right side of $x'\alpha'y'$ (see Fig. 5.6.b). This operation may be performed without introducing intersections by suitably shrinking the graph induced by $P$ and placing it close to $\alpha'$. Note that at this point the subgraph induced by $P$ and the subgraph induced by $Q$ are on the opposite sides of $x'\alpha'y'$, and that $y'$ can be connected to the vertex $z$ following $y'$ in $\tau$ crossing the boundary of the cluster of $y'$ only. Finally, observe that, since we have removed from $\sigma$ two monotonic sub-sequences, one increasing and one decreasing, with the same length, $Balance(\tau) = Balance(\sigma)$. □

Lemma 17 allows, for example, one to study the c-planarity of $cabcab$ instead of the c-planarity of $cabcabcab$ (by taking $\sigma_1 = c$, $x = a$, $\alpha = bc$, $y = a$, and $\sigma_2 = b$).
Lemma 18 Let $\sigma$ be a 3-cluster cycle. There exists a 3-cluster cycle $\tau$ such that: $\text{Balance}(\tau) = \text{Balance}(\sigma)$, $\tau$ is c-planar iff $\sigma$ is c-planar, and either $\tau$ is cyclically monotonic or $\tau = x\alpha y\beta$, where

1. $\alpha$ and $\beta$ are non-empty,
2. $x\alpha y$ is maximal monotonic increasing, and
3. $y\beta x$ is maximal monotonic decreasing.

Proof. If $\sigma$ is cyclically monotonic the statement is trivially true. If $\sigma$ is monotonic but not cyclically monotonic note that the length of $\sigma$ is at least 4. Suppose $\sigma = x_1x_2\alpha_1x_3x_4$, with $\alpha_1$ possibly empty, and suppose without loss of generality that $\sigma$ is monotonic increasing (otherwise $\sigma$ can be considered). Since all the subsequences of $\sigma$ are monotonic increasing and $x_4x_1x_2\alpha_1x_3$ is not monotonic, it follows that $x_4x_1$ is monotonic decreasing. Thus, Lemma 17 can be applied to $x_3x_4x_1x_2\alpha_1$, where $\sigma_1$ and $\alpha$ are empty, $x = x_3 = x_1$, $y = x_4 = x_2$, and $\sigma_2 = \alpha'$, obtaining the cycle $x_3x_4\alpha' = x_1x_2\alpha'$, which is cyclically monotonic increasing and which is c-planar iff $\sigma$ is c-planar.

Otherwise, suppose $\sigma$ is not monotonic. Sequence $\sigma$ is composed by $m \geq 2$ maximal monotonic sub-sequences. Namely, let $\sigma = x_1\alpha_1x_2\alpha_2\ldots x_m\alpha_m,$ where $x_i\alpha_ix_{i+1}$ is maximal monotonic and $\alpha_i$ is possibly empty ($x_{m+1} = x_1$). If $m = 2$, then, since $\sigma$ is not monotonic, both $\alpha_1$ and $\alpha_2$ are non-empty and the statement of the lemma holds.
If \( m > 2 \), by applying Lemma 17 we prove that there exists a sequence composed by \( m - 2 \) maximal monotonic subsequences that is c-planar iff \( \sigma \) is c-planar. By repeatedly applying this argument we find a sequence composed by one or two maximal monotonic subsequences for which one of the cases discussed above applies.

In order to reduce the number \( m \) of maximal monotonic sub-sequences by applying Lemma 17, assume that \( \alpha_i \) is one of the shortest of such sub-sequences (see Fig. 5.7) and consider the sub-sequence \( x_{i-1}\alpha_i x_i\alpha_{i+1} x_{i+1} \). Observe that it is possible to find in \( x_{i-1}\alpha_i x_i \) an \( x \) and in \( \alpha_{i+1} x_{i+1} \) a \( y \), such that \( x = x_{i+1} \), \( y = x_i \), and Lemma 17 can be applied where \( x' = x_i \), \( y' = x_{i+1} \), \( x'' = x_{i+1} \), \( y'' = y \), \( \alpha \) is the sequence of labels encountered traveling from \( x \) to \( x_i \) (end vertices excluded), and \( \overline{\alpha} = \alpha_i \) (see Fig. 5.7).

The following two lemmas (Lemma 19 and Lemma 20) study the c-planarity of the simple families of 3-cluster cycles cited in Lemma 18.

**Lemma 19** A 3-cluster cycle \( \sigma \) such that \( \sigma \) is cyclically monotonic is c-planar if and only if \( \text{Balance}(\sigma) = 3 \).

**Proof.** Since \( \sigma \) is monotonic we have that \( \text{Balance}(\sigma) \neq 0 \). Recall that \( \text{Balance}(\sigma) \) is a multiple of 3. If \( \text{Balance}(\sigma) = 3 \), then it can only be the case that \( \sigma = abc \) or \( \sigma = bca \) and \( \sigma = cab \) and it is trivial to see that \( \sigma \) is c-planar.

Suppose that \( \text{Balance}(\sigma) \geq 6 \). We show that \( \sigma \) is not c-planar. Suppose by contradiction that there exists a c-planar drawing \( \Gamma_\sigma \) of \( \sigma \). Since \( \text{Balance}(\sigma) \geq 6 \) each cluster has at least two vertices, and since the sequence is monotonic any three consecutive vertices of \( \sigma \) belong to three distinct clusters. Consider two vertices \( v_1 \) and \( v_2 \) belonging to the same cluster \( Z \) and such that it is possible to add an edge \((v_1, v_2)\) preserving the planarity of the drawing (see Fig. 5.8). Let \( p_1, n_1, p_2, \) and \( n_2 \) be the intersection points between the edges incident to \( v_1 \) and \( v_2 \) and the border of cluster \( Z \) in \( \Gamma_\sigma \), such that point \( p_1 \) (resp. \( p_2 \)) precedes \( v_1 \) (resp. \( v_2 \)) and point \( n_1 \) (resp. \( n_2 \)) follows \( v_1 \) (resp. \( v_2 \)) when moving along the cycle \( \sigma \). Since \( \sigma \) is monotonic, when moving from \( v_1 \) along \( \sigma \) and exiting cluster \( Z \) at point \( n_1 \), one reaches point \( p_2 \) before reaching point \( p_1 \). Analogously, when moving from \( v_2 \) along \( \sigma \) and exiting cluster \( Z \) at point \( n_2 \), one reaches point \( p_1 \) before reaching point \( p_2 \). A contradiction arises from the fact that the c-planarity of \( \sigma \) implies the planarity of a \( K_{3,3} \) graph with vertices \( \{v_1, p_2, n_2\} \) and \( \{v_2, p_1, n_1\} \). In fact, \( p_1 \) and \( p_2 \) (resp. \( n_1 \) and \( n_2 \)) can be connected with an edge along the border of \( Z \), and the path from \( n_1 \) to \( p_2 \) (resp. \( n_2 \) to \( p_1 \)) can be replaced with a single edge.
Figure 5.8: Illustration for the proof of Lemma 19. Triangular and square vertices show the subdivision of $K_{3,3}$.

**Lemma 20** Let $\sigma = x\alpha y\beta$ be a 3-cluster cycle, where $\alpha$ and $\beta$ are non-empty, $x\alpha y$ is maximal monotonic increasing, and $y\beta x$ is maximal monotonic decreasing. We have that $\sigma$ is c-planar iff $\text{Balance}(\sigma)$ is in $\{0, 3\}$.

![Diagram](image)

Figure 5.9: The construction of a c-planar drawing for a cycle $\sigma$ when $\text{Balance}(\sigma) = 0$ (a) and when $\text{Balance}(\sigma) = 3$ (b).

**Proof.** Let $\text{Balance}(\sigma) = 3k$, with $k$ a non-negative integer. Suppose $k$ is equal to 0 or 1. A c-planar drawing of $\sigma$ can be constructed by placing the vertices on three
half-lines as in the examples shown in Fig. 5.9.a and 5.9.b, respectively. The vertices of each half-line can be enclosed into a region representing their cluster.

Suppose that $k > 1$. We show that $\sigma$ is not c-planar. Suppose for a contradiction that $\sigma$ is c-planar and let $\Gamma_\sigma$ be a c-planar drawing of $\sigma$.

Since $\sigma$ is composed of two monotonic subsequences, there are exactly two “cusps” (vertices adjacent to two vertices of the same cluster). Therefore, there is at least a cluster $Z$ such that each vertex $z$ of $Z$ belongs to a sub-sequence $\rho = xzy$ or $\overline{\rho} = yzx$ of $\sigma$, with $x \neq y$. Since $k > 1$, the difference between the number of vertices of $Z$ belonging to a sub-sequence $\rho$ and the number of vertices of $Z$ belonging to a sub-sequence $\overline{\rho}$ is at least two (see Fig. 5.10). It follows that two vertices $v_1$ and $v_2$ can be identified such that they belong to a sub-sequence of the same kind ($\rho$ or $\overline{\rho}$) and such that it is possible to add an edge $(v_1, v_2)$ preserving the planarity of the drawing. A contradiction can be found by applying the same argument of Lemma 19 to $v_1$ and $v_2$.

Because of Lemma 18, Lemma 19, and Lemma 20, the problem of testing whether a 3-cluster cycle $\sigma$ is c-planar can be reduced to the problem of computing $\text{Balance}(\sigma)$. Since it is easy to compute $\text{Balance}(\sigma)$ in linear time (see Section 5.1), the following theorem holds.

**Theorem 15** Given an $n$-vertex 3-cluster cycle, there exists an algorithm to test if it is c-planar in $O(n)$ time.
In what follows we introduce a simple algorithm which guarantees the computation of a c-planar drawing of a 3-cluster cycle, if it admits one, in linear time. Consider a 3-cluster cycle $\sigma$ with $\text{Balance}(\sigma) \in \{0, 3\}$. Set a counter $C$ to zero. Visit $\sigma$ starting from the first vertex and adding (subtracting) one unit to $C$ when passing from $x$ to $y$, where $x \prec y$ ($y \prec x$). Remember that by convention a 3-cluster cycle is represented with a sequence $\sigma$ such that, when the vertices of the cycle are visited according to the order induced by $\sigma$ a non-negative value for $C$ is obtained. Without loss of generality we will further assume that $C$ never reaches a negative value. Otherwise, we can replace $\sigma$ with an equivalent cyclic permutation $\sigma'$ that has the above property. Permutation $\sigma'$ can be obtained in linear time from $\sigma$ by choosing as a starting vertex one for which the counter reaches the minimum value during the visit. Let $K$ be the maximum value assumed by $C$ during the visit.

We say that a vertex of $\sigma$ belongs to the $k$-th level iff $C$ has value $k$ when reaching such a vertex. The first vertex of $\sigma$ belongs to level 0. Note that each level contains vertices of the same cluster. Also, vertices belonging to level $k$ and level $k + 3$ belong to the same cluster. We denote with $\sigma|_k$ the sequence $\sigma$ restricted to level $k$, obtained from $\sigma$ by deleting all the vertices not belonging to the $k$-th level.

We construct a planar saturator in the following way. For each level $k \in \{0, \ldots, K\}$, we connect with an edge each pair of consecutive vertices of $\sigma|_k$. For each level $k \in \{0, \ldots, K - 3\}$, we insert an edge connecting the first vertex of $\sigma|_k$ with the last vertex of $\sigma|_{k + 3}$.

In order to show that the above defined saturator is planar we provide a c-planar drawing of the graph composed by the cycle and the saturator (see Fig. 5.11). First, we arrange all the vertices of $\sigma$ on a grid: the x-coordinate of a vertex is its position in $\sigma$ and the y-coordinate is its level. Then, we draw each edge of the cycle (excluding the one connecting the first and the last vertex of $\sigma$) with a straight segment without introducing intersections. Second, for each level $k \in \{0, \ldots, K\}$, we draw those edges of the saturator that connect pairs of consecutive vertices of $\sigma|_k$ with straight segments without introducing intersections. Note that, the sequence of the clusters at levels $0, \ldots, K - 3$ is the same as the sequence of the clusters at levels $3, \ldots, K$. Also, note that at this point of the construction, for each $k \in \{0, \ldots, K\}$ the first and the last vertices of $\sigma|_k$ are on the external face. Hence, the drawing can be completed without intersections by adding, for each level $k \in \{0, \ldots, K - 3\}$, the edge of the saturator connecting the first vertex of $\sigma|_k$ with the last vertex of $\sigma|_{k + 3}$ as shown in the example of Fig. 5.11. Finally, since the first and the last vertex of $\sigma$ are on the same face, they can be connected with a curve contained into such a face without introducing intersections.

It is easy to implement the above algorithm to work in linear time by building the lists of vertices for each level while visiting $\sigma$. Notice that $K$ is bounded by the
Figure 5.11: The construction of a c-planar drawing of a 3-cluster cycle \( \sigma \) in the case in which \( \text{Balance}(\sigma) = 3 \).

Hence, we can state the following result.

**Theorem 16** Given an \( n \)-vertex c-planar 3-cluster cycle \( \sigma \), there exists an algorithm that computes a c-planar drawing of \( \sigma \) in \( O(n) \) time.

From the above construction we also have the following.

**Theorem 17** A c-planar 3-cluster cycle admits a planar saturator that is the collection of three disjoint paths.

### 5.3 Clusters and Grammars

In this section we characterize the c-planar 3-cluster cycles in terms of formal grammars. Namely, we show that the sequences representing such cycles are those generated by a context-free grammar.
5. Cycles of Clusters

We denote by $\mathcal{L}$ the language of all strings on the alphabet $\{a, b, c\}$ such that each string:

(1) contains at least one instance of each label,

(2) does not contain repeated consecutive letters, and

(3) does not start and end with the same letter.

Observe that $\mathcal{L}$ describes all possible 3-cluster cycles. The following lemma holds:

**Lemma 21** $\mathcal{L}$ is a regular language.

**Proof.** The statement can be easily proved by showing that language $\mathcal{L}_1$ ($\mathcal{L}_2$, $\mathcal{L}_3$, respectively) of all the strings on the alphabet $\{a, b, c\}$, such that property 1 (2, 3, respectively) holds, is regular, and the intersection between regular languages is a regular language. In turn, the fact that $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ are regular languages can be easily proved by showing that they admit a regular expression. □

**Theorem 18** The following context-free grammar generates all and only the c-planar 3-cluster cycles:

\[
\begin{align*}
S & \rightarrow Z_0 | Z_3 \\
Z_0 & \rightarrow ABCB | ACBC | BCAC | BACA | CABA | CBAB \\
Z_3 & \rightarrow ABC | BCA | CAB \\
A & \rightarrow ABA | ACA | a \\
B & \rightarrow BAB | BCB | b \\
C & \rightarrow CAC | CBC | c
\end{align*}
\]

**Proof.** The proof exploits the same considerations used to prove Theorem 15. Note that symbol $Z_0$ generates all 3-cluster cycles $\sigma$ with $\text{Balance}(\sigma) = 0$ and symbol $Z_3$ generates all 3-cluster cycles with $\text{Balance}(\sigma) = 3$. □

**Theorem 19** The language of the c-planar 3-cluster cycles is not regular.
\textit{Proof.} The proof exploits the equivalence classes of the Myhill-Nerode theorem: given a language $L$ two strings $\alpha$ and $\beta$ are said to be equivalent if for each string $\gamma$, $\alpha\gamma$ and $\beta\gamma$ both belong or both do not belong to $L$. Language $L$ is regular iff the number of equivalence classes induced by the above equivalence relation is finite [HU79]. For each integer $n \geq 1$, denote by $\alpha_n$ the string $(abc)^n$. For each pair $n$, $m$, with $n < m$, $\alpha_n$ concatenated with $\gamma_n = (acb)^{n-1}$ yields a string of the language (corresponding to a c-planar cycle with balance 3) while $\alpha_m$ concatenated with $\gamma_n$ yields a string not belonging to the language (corresponding to a c-planar cycle with balance greater than 3). Thus, for $n < m$, $\alpha_n$ and $\alpha_m$ belong to two different equivalence classes. It follows that there is at least one equivalence class $[\alpha_n]$ for each $n \geq 1$ and thus the language of the c-planar 3-cluster cycles is not regular. \hfill \qed

From the above two theorems, we have that the language of the c-planar 3-cluster cycles is strictly context-free.

5.4 Cycles in Cycles of Clusters

![Figure 5.12: A clustered graph where at each level of the inclusion tree the nodes form a cycle. (a) A c-planar drawing. (b) The inclusion tree augmented with dashed edges that show the adjacencies between nodes at the same level.](image)
In this section we present a generalization of the results of Section 5.2. First, we generalize the results on 3-cluster cycles to k-cluster cycles. Second, we tackle the general problem of testing the c-planarity of a cycle that is clustered into a cycle of clusters that is in turn clustered into another cycle of clusters, and so on. An example is shown in Fig. 5.12. Fig. 5.12.a shows a c-planar clustered graph whose underlying graph is a cycle for which two levels of clusters are defined. Fig. 5.12.b puts in evidence the inclusion relationships between clusters of a given level and clusters of the level directly above it. The same figure shows also that the clusters of each level form a cycle (dashed edges).

We start by introducing preliminary assumptions and definitions. We consider clustered graphs $C(G,T)$ in which all the leaves of the inclusion tree $T$ have the same distance from the root. We call this distance the depth. A clustered graph which does not have this property can be easily reduced to this case by inserting “dummy” nodes in $T$. Hence, from now on we consider only inclusion trees whose leaves are all at the same depth. We define as $G_l(V_l,E_l)$ the graph whose vertices are the nodes of $T$ at distance $l$ from its root, and an edge $(\mu, \nu)$ exists if and only if an edge of $G$ exists incident to both $\mu$ and $\nu$.

For example, $G^0$ has only one vertex and $G^L$, where $L$ is the depth of the tree, is the underlying graph $G$ of $C(G,T)$. We label each vertex $v$ of $G^l$ with the cluster (corresponding to a vertex of $G^{l-1}$) to which $v$ belongs. If $G^l$ is a cycle, then it is possible to describe $G^l$ with the cyclic sequence of the labels of its vertices. If also $G^{l-1}$ is a cycle, we consider the labels of $G^l$ cyclically ordered according to the order they appear in $G^{l-1}$. Further, $Balance(G^l)$ can be defined as in Section 5.2 with values in $\{0, k, 2k, 3k, \ldots\}$, where $k$ is the length of $G^{l-1}$.

According to the above definitions a 3-cluster cycle is a clustered graph where $T$ has depth 2, $G^2$ is a cycle and $G^1$ is a cycle of length 3. In fact, the results of Section 5.2 can be extended to the case in which $G^1$ is a cycle of an arbitrary length.

**Theorem 20** Given an $n$-vertex clustered graph $C(G,T)$, such that $T$ has depth 2 and $G^1$ and $G^2$ are cycles, then:

1. there exists an algorithm to test if $C$ is c-planar in $O(n)$ time;
2. if $C$ is c-planar, a c-planar drawing of $C$ can be computed in $O(n)$ time.

**Proof.** The proof exploits the same considerations and constructions of Theorems 15 and 16. If the length of $G^1$ is $k$ then $C$ is c-planar iff $Balance(G^2) \in \{0, k\}$. In order to find a c-planar drawing of $C$, if it exists, the same strategy described in Section 5.2 can be applied, where, since in the construction depicted in Fig. 5.11 vertices belonging
Cycles in Cycles of Clusters

to level \( j \) and level \( j+k \) belong to the same cluster, an edge of the saturator is added between the first vertex of level \( j \) and the last vertex of level \( j+k \).

Let \( C(G,T) \) be a clustered graph and \( l \) be an integer between 1 and \( L \), where \( L \) is the depth of \( T \). A new clustered graph \( C^l(G,T^l) \) can be obtained from \( C \) by replacing \( T \) with a tree \( T^l \) obtained from \( T \) by connecting all the nodes at depth \( l \) with the root and deleting all the nodes having depth greater than zero and less than \( l \). According to this definition, \( C^1 = C \). The c-planarity of \( C^l \) can be used to study the c-planarity of \( C^{l-1} \), as is shown in the following lemma.

Lemma 22 Let \( C(G,T) \) be a clustered graph and \( l \) be an integer between 2 and \( L \), where \( L \) is the depth of \( T \). Let \( C^l \) be c-planar, \( G^l \) be a cycle, and \( G^{l-1} \) be a cycle of length \( k \). \( C^{l-1} \) is c-planar iff \( \text{Balance}(G^l) \in \{0,k\} \).

Proof. First, we prove that if \( \text{Balance}(G^l) \in \{0,k\} \), then \( C^{l-1} \) is c-planar by producing a planar drawing \( \Gamma_{C^{l-1}} \) of it. Since \( \text{Balance}(G^l) \in \{0,k\} \), then there exists a planar drawing \( \Gamma_{G^l} \) of \( G^l \) augmented with the edges of a planar saturator connecting vertices of \( G^l \) with the same label. The edges of this planar saturator are drawn in \( \Gamma_{G^l} \) either internally or externally with respect to the cycle \( G^l \) (see Fig. 5.13.a).

Let \( \Gamma_{C^l} \) be a planar drawing of the underlying graph \( G \) augmented with the edges of a planar saturator of \( C^l \). Such a drawing exists because \( C^l \) is c-planar. Two faces of \( \Gamma_{C^l} \) are incident to at least one vertex belonging to \( v \) for each vertex \( v \) of \( G^l \) (see Fig. 5.13.b). We call such faces \( f_i \) and \( f_e \), where \( f_e \) is the unbounded one. Also, we denote with \( v_{f_i,v} \) and \( v_{f_e,v} \) an arbitrary vertex of \( v \) incident to \( f_i \) and \( f_e \), respectively.

A saturator of \( C^{l-1} \) can be constructed from the planar saturator of \( C^l \) by adding one edge \( e' \) for each edge \( e \) of the planar saturator of \( G^l \) (see Figure 5.13.c). Edge \( e' \) is added within \( f_i \) if \( e \) is drawn internally (externally) in \( \Gamma_{G^l} \). Let \( v \) and \( \mu \) be the end vertices of \( e \). Suppose, without loss of generality, that \( e \) is added externally. Edge \( e' \) is attached to \( v_{f_e,v} \) and \( v_{f_e,\mu} \). The obtained saturator is planar since the starting drawing \( \Gamma_{C^l} \) is planar, and two edges of the saturator can not intersect in \( \Gamma_{C^{l-1}} \) since they don’t intersect in \( \Gamma_{G^l} \).

The second part of the proof shows that if \( C^{l-1} \) is c-planar then \( \text{Balance}(G^l) \in \{0,k\} \). Assume that there exists a planar saturator \( S \) of \( C^{l-1} \). Consider graph \( G^l \) obtained by adding to \( G \) the edges of the planar saturator \( S \). For each cluster \( \mu \) that is a vertex of \( G^l \) we contract all edges of \( G^l \) which have both extremes in \( \mu \) (we use label \( \mu \) to denote the resulting vertex) and remove multiple edges. Note that since the edges of \( S \) make each cluster connected we obtain a new graph \( G'' \) with the following properties:
5. CYCLES OF CLUSTERS

- there is a one-to-one correspondence among the vertices of $G^l$ and the vertices of $G''$,
- for each edge of $G^l$ there is a correspondent edge in $G''$.

Since the contraction operation preserves planarity and connectivity, the edges of $S$ that were incident to distinct clusters are still present in $G''$, such edges connect all vertices with the same label, and $G''$ is planar. Hence, the edges of $G''$ which have no corresponding edge in $G^l$ form a planar saturator for $G^l$. Since $G^l$ admits a planar saturator, its balance is in $\{0, k\}$. \hfill $\square$

**Lemma 23** Let $C = (G, T)$ be a clustered graph and let $l$ be an integer between 2 and $L$, where $L$ is the depth of $T$. If $C^l$ is not c-planar, then $C^1 = C$ is not c-planar.

**Proof.** If $C^l$ is not c-planar, any saturator introduces a subdivision of $K_{3,3}$ or $K_5$ in the graph $G$. Since any saturator of $C^l$ contains a (non-planar) saturator of $C^l$, it is always possible to find an obstruction in the graph $G$ augmented with the edges of the saturator of $C^l$. Hence $C^1$ can not be c-planar. \hfill $\square$

The following theorems state the main results about c-planarity testing for cluster graphs in which each $G^l$, $l \in \{1, \ldots, L\}$, is a cycle.

**Theorem 21** Given an $n$-vertex clustered graph $C(G, T)$, such that $T$ has depth $L$ and, for $l > 0$, $G^l$ is a cycle, there exists an algorithm to test if $C$ is c-planar in $O(Ln)$ time.

**Proof.** We apply Lemma 22 and Lemma 23 to the clustered graphs $C^l$ for $l = L, L - 1, \ldots, 2$. Since each test can be performed in $O(n)$ time, the statement follows. \hfill $\square$

**Theorem 22** Given an $n$-vertex clustered graph $C(G, T)$, such that $T$ has depth $L$ and, for $l > 0$, $G^l$ is a cycle, if $C$ is c-planar there exists an algorithm to compute a c-planar drawing of $C$ in $O(Ln)$ time.

**Proof.** Since Lemma 22 is proved by construction, by applying, level by level, Lemma 22 starting from level $L$ to level 1, a planar saturator of $C$ can be obtained. Since each step may be performed in $O(n)$ time, the statement follows. \hfill $\square$
Figure 5.13: (a) Drawing $\Gamma_{C^l}$ of $G^l$ with edges of the planar saturator added to the external or internal face. (b) Drawing $\Gamma_{C^l}$ in which two faces (called the internal and external face) touching all the clusters can be found. (c) saturator edges joining suitable vertices of $\Gamma_{C^l}$.
Chapter 6

Clustered Cycles

In this chapter we propose a polynomial time c-planarity testing and embedding algorithm for rigid clustered cycles. Rigid clustered cycles are flat clustered graph whose underlying graph is a cycle and whose graph of the clusters has a fixed planar embedding.

6.1 The Problem

Let $\Gamma$ be a planar drawing of a planar graph $G$ and $c$ be a cycle composed of vertices and edges of $G$. We deal with the problem of testing if $c$ can be drawn on $\Gamma$ without crossings.

Of course, if the vertices of $G$ are drawn as points, the edges as simple curves, and the drawing of $c$ must coincide with the drawing of its vertices and edges, then the problem is trivial. In this case $c$ can be drawn without crossings if and only if it is simple.

We consider the problem from a different point of view. Namely, we suppose that the vertices of $G$ are drawn in $\Gamma$ as “small circles” and the edges as “thin stripes”. Hence, $c$ can pass several times through a vertex or through an edge without crossing itself. In this case even a non-simple cycle can have a chance to be drawn without crossings. For example, the cycles of Figs. 6.1.a and 6.1.c can be drawn without crossings, while the cycles of Figs. 6.1.b and 6.1.d cannot.

The problem, in our opinion, is interesting in itself. However, we study it because of its meaning in the field of clustered planarity.

We define as clustered cycle a flat clustered graph whose underlying graph is a cycle. A rigid clustered cycle is a clustered cycle $C$ in which $G^1(C)$ has a prescribed
Figure 6.1: Examples of cycles which can be drawn without crossings ((a) and (c)) and which cannot ((b) and (d)).
planar embedding. This is obviously the case when $G^1(C)$ is triconnected. In this chapter we tackle the c-planarity testing and embedding problem for rigid clustered cycles. Namely, consider again the problem stated at the beginning of this section and the examples of Fig. 6.1 according to the above definitions. The cycle is the underlying graph of a flat clustered graph and the nodes of the graph are the clusters. Thus, the instance depicted in Fig. 6.1.a has four clusters and an underlying graph which is a cycle of 12 vertices. If you are able to find a drawing of the cycle without intersections you are also able to find a c-planar embedding for the rigid clustered cycle and vice versa. Another description of the c-planarity testing on rigid clustered cycles can be found in [CDMP05].

### 6.2 Basic Definitions

In the following we need a slightly wider definition of clustered cycle in which $G^1(C)$ is allowed to have multiple edges between two nodes. In fact, our planarity testing algorithm is based on a sequence of transformations of the input instance, in which graph $G^1(C)$ may temporarily become a multigraph.

We define a *clustered cycle* $C(G, G^1, \Phi_V, \Phi_E)$, where $G^1$ is a graph, possibly with multiple edges, $G$ is a cycle, $\Phi_V$ maps each vertex of $G$ to a vertex of $G^1$, and $\Phi_E$ maps each edge of $G$ between vertices $v_1 \in \mu_1$ and $v_2 \in \mu_2$, where $\mu_1 \neq \mu_2$, to an edge of $G^1$ between vertices $\mu_1$ and $\mu_2$.  

In the following, to avoid ambiguities, the graph $G^1$ of a clustered cycle $C$ will be denoted as $G^1(C)$, its edges will be called *pipes* while its vertices will be called *nodes* or *clusters*. For example, the instance of Fig. 6.1.a has four clusters and five pipes. Also, without loss of generality, we will consider instances where $G^1(C)$ has no empty pipe.

Given a cluster $\mu \in G^1(C)$, we denote by $\text{deg}(\mu)$ the number of pipes that are adjacent to $\mu$ in $G^1(C)$, where multiple pipes count for their multiplicity. The *size* of a pipe of $G^1(C)$ is the number of edges of $G$ it contains. As an example, the leftmost cluster of Fig. 6.1.a has degree three, and the central pipe has size two.

It is easy to see that a path in $G$ whose vertices belong to the same cluster can be collapsed into a single vertex without affecting the c-planarity property of the clustered cycle. Hence, in the following we consider only clustered cycles where consecutive vertices belong to distinct clusters. We call *cusp* a vertex $v$ of $G$ whose incident edges $e_1$ and $e_2$ are such that $\Phi_E(e_1) = \Phi_E(e_2)$.

A rigid clustered cycle is such that the embedding of $G^1(C)$ is specified. Given a rigid clustered cycle $C$ the *embedding* $\Lambda$ of $C$ is the specification, for each pipe $a$ in $G^1(C)$ and for each end node $\mu$ of $a$, of the total ordering $\lambda_\mu(a)$ of the edges contained...
in $a$ when turning around $\mu$ clockwise. An embedding of a clustered cycle is c-planar if there exists a planar drawing of $C$ that respects such an embedding. If an embedding is c-planar, for each pipe $a = (\mu, \nu)$, we have that $\lambda_{\mu}(a) = \lambda_{\nu}(a)$, where $\lambda_{\nu}(a)$ denotes the reverse of $\lambda_{\nu}(a)$. Hence, to describe a c-planar embedding it is sufficient to specify for each pipe the order of its edges with respect to one of its end nodes only.

### 6.3 Fountain Clusters

Consider a clustered cycle $C$ and one of its clusters $\mu = \{v_1, \ldots, v_q\}$. Let $e'_i$ and $e''_i$ be the incident edges of $v_i$. Cluster $\mu$ is a fountain cluster if there exists a pipe $b$, called base of $\mu$, such that for each $v_i$ we have that $e'_i \in b$ or $e''_i \in b$ (see Fig. 6.2 for an example). A fountain clustered cycle is a clustered cycle in which each cluster is a fountain cluster.

Let $\mu$ be a fountain cluster and let $b$ be a base of $\mu$. The following properties hold:

**Property 10** The edges incident to a cusp $v$ of $\mu$ belong to $b$.

**Property 11** Cluster $\mu$ has a base $b' \neq b$ if and only if $\deg(\mu) = 2$ and no cusp belongs to $\mu$. Otherwise $\mu$ has exactly one base.

**Property 12** Let $p \neq b$ be a pipe incident to $\mu$. If $p$ is a base for $\mu$ then $\text{size}(p) = \text{size}(b)$, otherwise $\text{size}(p) < \text{size}(b)$. 
The following property implies that the c-planar embedding of a fountain clustered graph is completely described by the order of the edges of the bases.

**Property 13** Let $\Lambda$ be a c-planar embedding of a clustered cycle $C$, and let $\mu$ be a fountain cluster with base $b$. For any pipe $a \neq b$ we have that $\lambda_\mu(a)$ is $\lambda_\mu(b)$ restricted to the edges adjacent to the edges of $a$.

**Cluster Expansion**

Given a cluster $\mu$ of a clustered cycle $C$, we call cluster expansion of $\mu$ the following operation that produces the clustered cycle $C'$ (see Fig. 6.3.)

Let $a_1, \ldots, a_k$ be the pipes incident to $\mu$, where $k = \deg(\mu)$. Cluster $\mu$ is replaced in $C'$ with $k$ new clusters $\mu_1, \ldots, \mu_k$, each one incident to pipes $a_1, \ldots, a_k$, respectively. All the other clusters of $C$ stay unchanged in $C'$. Each non-cusp vertex $v$ in $\mu$ incident to edges $e_i \in a_i$ and $e_j \in a_j$ is replaced in $C'$ by two new vertices $v'$ and $v''$, with $v' \in \mu_i$ and $v'' \in \mu_j$. A new pipe $(\mu_i, \mu_j)$ is inserted, if not already present, and a new edge $(v', v'')$ is added such that $(v', v'') \in (\mu_i, \mu_j)$. Each cusp vertex $v$ having its edges in pipe $a_i$ stays unchanged in $C'$ and is assigned to cluster $\mu_i$.

Now, we investigate the properties of the cluster expansion operation. Let $C'$ be a clustered cycle obtained from $C$ by applying a cluster expansion on cluster $\mu$, let $a_1, \ldots, a_k$ be the pipes incident to $\mu$, and let $\mu_1, \ldots, \mu_k$ the corresponding clusters introduced by the cluster expansion.

**Property 14** Each cluster $\mu_i$ produced by the cluster expansion is a fountain cluster with base $a_i$.

**Property 15** There are no multiple pipes incident to the clusters $\mu_i$ produced by a cluster expansion.

By Property 15 cluster expansion can be used to eliminate multiple pipes incident to $\mu$.

**Property 16** Let $n_\mu$ be the number of vertices of $\mu$ and let $n_{\mu_i}$ be the number of vertices of $\mu_i$. We have that $\sum_i n_{\mu_i} \leq 2n_\mu$.

**Property 17** The cluster expansion operation on cluster $\mu$ can be performed in $O(n_\mu)$ time, where $n_\mu$ is the number of vertices of $\mu$.

**Property 18** Applying a cluster expansion to each non-fountain cluster of $C$ produces a fountain clustered cycle.
6. Clustered Cycles

Figure 6.3: An example of cluster expansion: (a) A non-fountain cluster $\mu$. (b) The result of the cluster expansion.

Up to now, the expansion operation has been defined without considering the embedding of $G^1(C)$ and $G^1(C')$. If $C$ is a rigid clustered cycle it is easy to extend the definition of cluster expansion considering also embedding issues. The general idea is to embed the pipes around nodes $\mu_i$ respecting the order that pipes $a_i$ had in $C$ around $\mu$. Namely, consider, without loss of generality, cluster $\mu_1$ inserted by the cluster expansion. In addition to pipe $a_1$, the pipes incident to $\mu_1$ are a subset of $\{(\mu_1, \mu_2), \ldots, (\mu_1, \mu_k)\}$ and are embedded around $\mu_1$ as pipes $\{a_1, \ldots, a_k\}$ were embedded around $\mu$.

Note that, even if the embedding of $G^1(C)$ is planar, the obtained embedding of $G^1(C')$ may be not planar due to the pipes $(\mu_i, \mu_j)$ inserted among the clusters $\mu_1, \ldots, \mu_k$. Given a rigid clustered cycle $C$, a cluster expansion of one of its clusters $\mu$ is feasible if the obtained embedding of $G^1(C')$ is planar, that is, if $C'$ is a rigid clustered cycle. Examples of feasible and non-feasible cluster expansions are shown in Fig. 6.4.

Lemma 24 Given a rigid clustered cycle $C$, if a cluster expansion of one of its clusters $\mu$ is not feasible, then $C$ is not c-planar.

Proof. Consider the circular ordering of cluster $\mu_1, \ldots, \mu_k$ induced by the circular ordering of pipes $a_1, \ldots, a_k$ around $\mu$. Denote by $\mu_i < \mu_j < \mu_k$ the fact that $\mu_i, \mu_j,$
and $\mu_h$ are encountered in this order in the circular ordering. Since the embedding of $G^1(C)$ is planar and the embedding of $G^1(C')$ is not, there must be two pipes $(\mu_i, \mu_h)$ and $(\mu_j, \mu_l)$ in $G^1(C')$ such that $\mu_i \prec \mu_j \prec \mu_h \prec \mu_l$. This implies that there exist in $C$ two paths of $G$, one traversing $a_i, \mu, a_h$ and the other traversing $a_j, \mu, a_l$. Since the embedding of $\mu$ is fixed, these two paths cannot be drawn without intersections. Therefore $C$ is not $c$-planar.

\begin{lemma}
Let $C$ be a rigid clustered cycle and let $\mu$ be a cluster of $C$. Let $C'$ be the result of a feasible cluster expansion applied to $\mu$. $C$ is $c$-planar if and only if $C'$ is $c$-planar.
\end{lemma}

\begin{proof}
Suppose that $C$ is $c$-planar, and let $\Gamma$ be a $c$-planar embedding of $C$. A $c$-planar embedding $\Gamma'$ of $C'$ can be computed as follows. For each pipe that is present both in $C$ and in $C'$, including pipes $a_1, \ldots, a_k$ incident to $\mu$, we assume that the order of edges in $\Gamma'$ is the same as in $\Gamma$. The order of the edges inside the pipes added among nodes $\mu_1, \ldots, \mu_k$ is determined, according to Properties 14 and 13, by the their order in the bases $a_1, \ldots, a_k$. Hence, the $c$-planarity of $\Gamma'$ follows from the $c$-planarity of $\Gamma$.

Suppose now that $C'$ is $c$-planar, and let $\Gamma'$ be a $c$-planar embedding of $C'$. A $c$-planar embedding $\Gamma$ of $C$ can directly obtained from $\Gamma'$. Since all pipes of $C$ are also present in $C'$, the order of their edges can be assumed to be the same as in $\Gamma'$.

Figure 6.4: The result of a feasible (a) and a non-feasible (b) cluster expansion.
6. Clustered Cycles

Consider edge $e$ of pipe $(\mu_i, \mu_j)$ in $\Gamma'$. The path $e_i, e, e_j$ of $\Gamma'$, where $e_i \in a_i$ and $e_j \in a_j$ corresponds to path $e_i, e_j$ in $\Gamma$. Hence, the c-planarity of $\Gamma'$ implies the c-planarity of $\Gamma$. 

Pipe Contraction

We call a pipe $b$ between two fountain clusters $\mu$ and $\nu$ contractible if:

- $b$ is the only pipe between $\mu$ and $\nu$,
- $b$ is a base for both $\mu$ and $\nu$, and
- $b$ is the only base for one of them.

We define the pipe contraction operation on a contractible pipe $b$ as follows. The pipe contraction produces a clustered cycle $C'$ starting from a clustered cycle $C$ by replacing $\mu, \nu$, and $b$, with a new cluster $\mu'$, which is adjacent to all the clusters which $\mu$ and $\nu$ were adjacent to. If $\mu$ and $\nu$ were adjacent to the same cluster $\rho$, $\mu'$ is doubly adjacent to $\rho$; that is, the pipe contraction may introduce multiple pipes incident to $\mu'$. An example of pipe contraction is shown in Fig. 6.5. Note that the new cluster $\mu'$ is, in general, not a fountain cluster.

![Figure 6.5: An example of pipe contraction: (a) pipe $b$ before contraction; (b) The result of the contraction of $b$.](image)

Since $b$ is a base for both $\mu$ and $\nu$, each edge $e_m$ entering $\mu$ or $\nu$ belongs to a path $p_C = e_m, v, e_1, v_1, \ldots, e_k, v_k, e_{out}$, where $e_{out}$ is the first edge exiting $\mu$ or $\nu$ and $e_i \in b$, $i = 1, \ldots, k$, $k \geq 1$. Path $p_C$ is replaced by $p_{C'} = e_{in}, v_{\mu'}, e_{out}$, with $v_{\mu'} \in \mu'$. 

86
If $C$ is a rigid clustered cycle, analogously to the expansion operation, the definition of the pipe contraction operation can be extended in order to take into account embedding issues. Namely, we can give to the pipes around $\mu'$ the same circular order they have in $C$ around the subgraph composed of $\mu$, $\nu$, and $b$.

The following properties are trivial to prove.

**Property 19** Let $n_{\mu}$ and $n_{\nu}$ be the number of vertices of $\mu$ and $\nu$, respectively. The number of vertices of $\mu'$ is at most $(n_{\mu} + n_{\nu})/2$.

**Property 20** The pipe contraction operation on a contractible pipe $b$ can be performed in $O(size(b))$ time.

![Figure 6.6: A drawing $\Gamma'$ of $C'$ (a) and the corresponding drawing $\Gamma$ of $C$ (b).](image)

**Lemma 26** Let $C$ be a rigid fountain clustered cycle and let $b$ be a contractible pipe. Let $C'$ be the rigid clustered cycle obtained from $C$ by applying a pipe contraction operation to $b$. $C$ is c-planar if and only if $C'$ is c-planar.

**Proof.** Suppose that $C$ is c-planar, let $\Gamma$ be a c-planar drawing of $C$, we show how to build a c-planar drawing $\Gamma'$ of $C'$ by slightly modifying $\Gamma$. Namely, region $R(\mu')$ is the union of $R(\mu)$, $R(\nu)$, and the stripe corresponding to $b$. (Observe that $R(\mu')$
is connected.) Each path \(p_C = e_{in}, v, e_1, v_1, \ldots, e_k, v_k, e_{out}\) of \(C\), with \(\Phi_E(e_i) = b\), is replaced by \(p'_C = e_{in}, v_{i'}, e_{out}\), where \(v_{i'}\) replaces \(v\), and all vertices \(v_i\), with \(i = 1, \ldots, k\), are removed, joining their incident edges. It is easy to see that the obtained drawing is a c-planar drawing of \(C'\).

Suppose now that \(C'\) is c-planar, and let \(\Gamma'\) be a c-planar drawing of \(C'\). We provide a c-planar drawing \(\Gamma\) of \(C\) by suitably modifying \(\Gamma'\). We take region \(R(\mu) = R(\mu')\). Observe that in \(\Gamma'\) all the pipes that were incident to \(v\) are consecutively attached to the border of \(R(\mu')\). Hence, it is possible to add two arbitrarily thin stripes, corresponding to \(b\) and \(R(v)\), respectively, along the border of \(R(\mu')\) in such a way to intersect those pipes only (see Fig 6.6.b).

Now, consider the edges entering \(R(\mu')\) that were incident to \(\mu\) before contraction in counterclockwise order. Let \(e_{in}\) be the current edge and \(p_C = e_{in}, v_{i'}, v_{i'}, \ldots, e_k, v_k, e_{out}\) be the path of \(C'\) that replaced \(p_C = e_{in}, v, e_1, v_1, \ldots, e_k, v_k, e_{out}\). (Remember that \(k \geq 1\).)

If \(k = 1\), it is easy to obtain a drawing of \(p_C = e_{in}, v, e_1, v_1, e_{out}\) starting from the drawing of \(p_C = e_{in}, v_{i'}, e_{out}\) by replacing \(v_{i'}\) with \(v\) and splitting \(e_{out}\) with a vertex \(v_1\) in such a way that \(v_1\) is into \(R(\mu)\) (see paths \(p_C^{1}\) and \(p_C^{1}\) of Fig 6.6 for an example.)

Analogously, if \(k\) is odd \((e_{out} = e_{in})\) it is possible to draw \(p_C = e_{in}, v, e_1, v_1, \ldots, e_k, v_k, e_{out}\) in a thin stripe along the drawing of \(p_C = e_{in}, v_{i'}, e_{out}\) (see paths \(p_C^{odd}\) and \(p_C^{odd}\) of Fig 6.6 for an example).

If \(k\) is even, then both \(e_{in}\) and \(e_{out}\) were incident to \(\mu\) in \(C\). In this case the drawing of \(p_C = e_{in}, v_{i'}, e_{out}\) does not immediately provide a drawing of \(p_C = e_{in}, v, e_1, v_1, \ldots, e_k, v_k, e_{out}\), which can be built as follows. Vertex \(v\) is placed into \(R(\mu)\) as edge \(e_{in}\) crosses the border of \(R(\mu)\). Edge \(v_1\) follows clockwise the border of \(R(\mu)\) till the previous edge \(e_{in}\) entering \(R(\mu)\) is found (or \(R(v)\) is reached). Since edges \(e_{in}\) are considered in counterclockwise order and since \(b\) was a base for both \(\mu\) and \(v\), path \(p'_C\), starting with edge \(e_{in}\), always has vertex \(v'\) into \(R(\mu)\) and \(v'_1\) into \(R(v)\). Therefore, edge \(e_1\) can be drawn arbitrarily near to path \(p'_C\) and can be terminated with \(v_1\) placed into \(R(v)\). Edges \(e_i\), with \(i = 2, \ldots, k\), can be drawn in an arbitrarily thin stripe adjacent to \(e_1\), positioning \(v_i\) alternately into \(R(\mu)\) and \(R(v)\). Finally, edge \(e_{out}\) can follow path \(p_C\) to exit \(R(v)\) (see paths \(p_C^{even}\) and \(p_C^{even}\) of Fig 6.6 for an example.)

Now, consider the edges entering \(R(\mu')\) that were incident to \(v\) before the contraction. Let \(e_{in}\) be the current edge and \(p_C = e_{in}, v_{i'}, v_{i'}, \ldots, e_k, v_k, e_{out}\) be the path of \(C'\) that replaced \(p_C = e_{in}, v, e_1, v_1, \ldots, e_k, v_k, e_{out}\). If \(k\) is odd, the drawing of \(p_C = e_{in}, v, e_1, v_1, \ldots, e_k, v_k, e_{out}\) was already considered above. If \(k\) is even, then the whole path \(p_C\) can be drawn in an arbitrary small stripe along \(e_{in}\) (see paths \(p_C^{even}\) and \(p_C^{even}\) of Fig 6.6 for an example.)
6.4 C-Planarity Testing of Clustered Cycles

In this section we describe a c-planarity testing algorithm for rigid clustered cycles. The algorithm is based on the following lemmas.

**Lemma 27** Let $C$ be a fountain clustered cycle such that $G^1(C)$ is not a simple cycle and has not multiple pipes. There exists at least one contractible pipe $b^*$ in $G^1(C)$.

![Diagram](image)

Figure 6.7: Three cases used in the proof of Lemma 27: (a) cluster $\mu_{i+1}$ has degree different from two; (b) $\mu_{i+1}$ has degree two and $b_i$ is the only base for $\mu_{i+1}$; and (c) $\mu_{i+1}$ has degree two and has two bases.

**Proof.** We search for a contractible pipe by considering one pipe at a time. We start from a pipe $b_1(\mu_1, \mu_2)$ of maximum size in the whole clustered cycle $C$. Observe that,
since $b_1$ is a pipe of maximum size also among those incident to both $\mu_1$ and $\mu_2$, by Property 12, $b_1$ is a base for both of them. If $\mu_1$ has degree different from two, by Property 11, $\mu_1$ has a single base and the statement holds with $b^* = b_1$. If $\mu_1$ has degree two and its second pipe $b_0$ is not a base for $\mu_1$, then again the statement holds with $b^* = b_1$. Otherwise, let $b_i(\mu_i, \mu_{i+1})$ be the current pipe, where $b_i$ is a base for both $\mu_i$ and $\mu_{i+1}$ and $\mu_i$ has a second base $b_{i-1}$. Two cases are possible:

1. $\mu_{i+1}$ has degree different from two (see Fig. 6.7.a). In this case, by Property 11, $\mu_{i+1}$ has the single base $b_i$ and the statement holds with $b^* = b_i$.

2. $\mu_{i+1}$ has degree two.
   a) If $\mu_{i+1}$ only admits $b_i$ as a base (see, for example, Fig. 6.7.b), then again the statement holds with $b^* = b_i$.
   b) If $\mu_{i+1}$ has two bases (see, for example, Fig. 6.7.c), let $b_{i+1} = (\mu_{i+1}, \mu_{i+2})$ be the second base of $\mu_{i+1}$. Due to Property 12, $size(b_{i+1}) = size(b_i)$. Therefore $b_{i+1}$ is also a base for its incident cluster $\mu_{i+2} \neq \mu_{i+1}$. We carry on the search by taking as current pipe $b_{i+1}$.

Since $G^1(C)$ is not a simple cycle, it is assured that the current pipe $b_i$ is different from the starting pipe $b_1$ and that there exists at least a $j$ for which $b_j$ is the only base for $\mu_j$.

We introduce a quantity, denoted $E(C)$, that will be used to analyze the algorithm both in terms of correctness and in terms of time complexity. Intuitively, it is an indicator of the structural complexity of $G^1(C)$. Quantity $E(C)$ is defined as follows:

$$E(C) = \sum_{a \in \{\text{pipes of } G^1(C)\}} (size(a))^2.$$  

Our c-planarity testing algorithm consists of a sequence of transformations of the input clustered cycle $C$. Some of these transformations are pairs of consecutive contraction-expansion operations, that is, we select a contractible pipe $b$, we contract $b$ generating a new cluster $\mu$, and we perform a cluster expansion on $\mu$. The following lemmas show how $E$ changes after a pair of consecutive contraction-expansion operations. First, we need this technical lemma:

**Lemma 28** Let $C$ be a fountain clustered cycle and let $b = (\mu', \mu'')$ be a contractible pipe. Let $C'$ be the clustered cycle obtained by applying a pipe contraction to $b$ producing cluster $\mu$. Let $C^*$ be the clustered cycle obtained by applying a cluster expansion to $\mu$ which is replaced in $C^*$ by clusters $\mu_1, \ldots, \mu_k$. If one between clusters $\mu_1, \ldots, \mu_k$ has degree one then $b$ contains edges incident to cusps.
C-Planarity Testing of Clustered Cycles

Proof. Suppose cluster $\mu_j$ has degree one. Vertices in $\mu_j$ are cusps. Now, consider pipe $a_j$ incident to $\mu_j$. Edges contained in $a_j$ are all adjacent to the cusps contained into $\mu_j$. Pipe $a_j$ is also present in $C$, where it is incident to $\mu'$ or $\mu''$ (say $\mu'$) and has the same internal edges. It is easy to show that $a_j$ cannot contain edges incident to cusps in $C$. In fact, two cases are possible:

- $a_j$ is a base for $\mu'$. In this case, by Property 11, $\mu'$ has degree two and cannot have cusps.
- $a_j$ is not a base for $\mu'$. In this case, Property 10 guarantees that $a_j$ cannot contain edges incident to cusps.

Therefore, edges in $a_j$ must be part of paths traversing $b$ more than one time. This proves the statement. 

Lemma 29 Let $C$ be a fountain clustered cycle and let $b = (\mu', \mu'')$ be a contractible pipe. Let $C'$ be the clustered cycle obtained by applying a pipe contraction to $b$ producing cluster $\mu$. Let $C^*$ be the clustered cycle obtained by applying a cluster expansion to $\mu$. If $C^*$ has only one pipe $p$ more than $C'$, i.e., the same number of pipes of $C$, then $\mathcal{E}(C^*) < \mathcal{E}(C)$.

Proof. In order to prove the statement, it is sufficient to show that $\text{size}(b) > \text{size}(p)$. We prove this by showing that each edge of $p$ corresponds to one edge of $b$, while $b$ contains at least one extra edge for each cusp of $\mu'$ and $\mu''$, which have at least one cusp.

By construction, $p$ can not have edges incident to cusps. Also, by construction each edge $e$ in $p$ corresponds to a path traversing $b$. Edge $e$ corresponds to a single edge of $b$ if such a path has no cusp and to more than one edge otherwise. Therefore, in order to prove that $\text{size}(b) > \text{size}(p)$ it is sufficient to show that $b$ has edges incident to cusps. In fact, suppose by contradiction that $b$ has no edge incident to a cusp.

- If one between $\mu'$ and $\mu''$, say $\mu'$, has degree one, then $\mu'$ contains only cusps, contradicting the fact that $b$ does not contains edges incident to cusps.
- If both $\mu'$ and $\mu''$ have degree two, then, since $b$ is a base for both of them, by Property 11 $b$ can not be the single base for one of them, contradicting the fact that $b$ is a contractible pipe.
- Otherwise, one between $\mu'$ or $\mu''$, say $\mu'$, has degree greater than two and $\mu''$ has degree at least two. In this case, the cluster expansion applied to $\mu$ inserts clusters $\mu_1, \ldots, \mu_k$, with $k > 2$. Therefore, there must be one cluster $\mu_j$ which

91
6. Clustered Cycles

had degree one and by Lemma 28 we have that $b$ contains edges incident to cusps.

From the above discussion we have that $size(b) > size(p)$ and, hence, $E(C^*) < E(C)$.

Lemma 30 Let $C$ be a fountain clustered cycle and let $b = (\mu', \mu'')$ be a contractible pipe. Let $C^*$ be the clustered cycle obtained by applying a pipe contraction to $b$ followed by a cluster expansion of the obtained cluster $\mu$. We have that $E(C^*) < E(C)$.

Proof. Let $C'$ be the clustered cycle generated by the pipe contraction applied to $b = (\mu', \mu'')$, where, without loss of generality, $b$ is the only base for $\mu'. C'$ contains all the pipes of $C$ with the exception of $b$, then $E(C') = E(C) - (size(b))^2$.

$C^*$ has the same pipes of $C'$ plus a set of new pipes $p_1, \ldots, p_h$. If $h = 0$ then $E(C^*) = E(C') < E(C)$. If $h = 1$ by Lemma 29 $E(C^*) < E(C)$.

Suppose $h \geq 2$. We have that $E(C^*) = E(C') + \sum_{j=1}^{h} (size(p_j))^2 = E(C) - (size(b))^2 + \sum_{j=1}^{h} (size(p_j))^2$. Observe that each edge contained in the pipes $p_1, \ldots, p_h$ is generated by the split of a vertex in $\mu$, and that by construction the number of vertices in $\mu$ is at most $size(b)$. Then, $\sum_{j=1}^{h} size(p_j) \leq size(b)$. Hence, $\sum_{j=1}^{h} (size(p_j))^2 < (size(b))^2$, and the statement follows.

Lemma 31 A clustered cycle $C$ whose graph of the clusters $G^1(C)$ is a path is c-planar.

Proof. Let $\mu_1, \ldots, \mu_m$ be the nodes of $G^1(C)$ in the order in which they appear in the path. A planar embedding of $C$ can be built as follows. Traverse the cycle $G$ starting from a vertex in $\mu_1$. Each edge $e$ belonging to pipe $a = (\mu_i, \mu_j)$ is inserted at the last position of $\lambda_{\mu_i}(a)$ and at the first position of $\lambda_{\mu_j}(a)$. When the path comes back to $\mu_1$ for the last time it can be connected to the starting point preserving c-planarity. See Fig. 6.8 for an example.

We are now ready to introduce the c-planarity testing algorithm for a rigid clustered cycle $C$. First, the algorithm performs a cluster expansion for each cluster. If one of such expansions is not feasible, then, according to Lemma 24, $C$ is not c-planar. If all the expansions are feasible, according to Property 18, we obtain a fountain clustered cycle $C^f$, which is c-planar if and only if $C$ is c-planar. Also, by Property 15, $C^f$ does not have multiple pipes. If $G^1(C^f)$ is a cycle, then the c-planarity of $C^f$ can be easily tested using the results described in Chapter 5 (see also [CDPP04, CDPP05]). If $G^1(C^f)$ is a path, then Lemma 31 states that $C^f$ is c-planar.
C-Planarity Testing of Clustered Cycles

Figure 6.8: A c-planar drawing of clustered cycle $C$ whose graph of the clusters $G^1(C)$ is a path.

If $G^1(C^f)$ is not a cycle nor a path, then Lemma 27 guarantees that there exists a contractible pipe $b^* = (\mu, \nu)$. Perform a contraction operation on $b^*$. Perform a cluster expansion on the resulting cluster. Note that the pipe contraction may temporarily generate multiple pipes; however, by Property 15, the subsequent cluster expansion produces a clustered cycle which has no multiple pipes.

If the expansion is not feasible, then $C$ is not c-planar (Lemma 24). Otherwise, we obtain a fountain clustered cycle with no multiple pipes which is c-planar if and only if $C$ is c-planar and we can iterate the two last operations until the clusters of the clustered cycle form a cycle, or a path, or a cluster expansion is non-feasible.

The algorithm, called ClusteredCyclePlanarityTesting, is formally described in Fig. 6.9.

**Theorem 23** Given a rigid clustered cycle $C(G, G^1, \Phi_V, \Phi_E)$, there exists an algorithm to test if $C$ is c-planar in $O(n^3)$ time, where $n$ is the number of vertices of $G$.

**Proof.** First, we prove that algorithm ClusteredCyclePlanarityTesting gives the correct result. If all initial cluster expansions are feasible, then, by Lemma 25 and Properties 18 and 15, we obtain a fountain clustered cycle without multiple edges which is c-planar if and only if the input clustered cycle $C$ is c-planar. Otherwise, if a non-feasible cluster expansion is encountered, we have by Lemma 24 that $C$ is not c-planar.

Let $C_i$ be the current fountain clustered cycle with no multiple edges. If $G^1(C_i)$ is a cycle, then the c-planarity testing algorithm for k-cluster cycles shown in Chapter 5 can be applied. If $G^1(C_i)$ is a path, by Lemma 31, it is always c-planar. Otherwise, Lemma 27 guarantees that there exists a contractible pipe $b^*$ in $C_i$. A pair of pipe contraction and cluster expansion can be performed producing a fountain clustered cycle $C_{i+1}$ with no multiple edges which, by Lemmas 25 and 26, is c-planar if and only if $C_i$ is c-planar. By Lemma 30, each pair of pipe contraction and cluster expansion operations decreases $E$. Since $E$ can not be negative, the body of the while cycle is executed a finite number of times and therefore the algorithm always terminates.
Algorithm \textit{ClusteredCyclePlanarityTesting}

\textbf{input} A rigid clustered cycle $C$

\textbf{output} True if $C$ is c-planar, false otherwise

\begin{verbatim}
for all clusters $\mu$ in $C$ do
    perform a cluster expansion of $\mu$
    if the cluster expansion of $\mu$ is not feasible then
        return false
    end if
end for
{at this point $C$ is a fountain clustered cycle with no multiple pipes}

while $C$ is not a cycle or a path do
    let $b$ be a contractible pipe of $C$
    apply a pipe contraction to $b$, obtaining cluster $\mu'$.
    perform a cluster expansion of $\mu'$
    if the cluster expansion of $\mu'$ is not feasible then
        return false
    end if
end while
{at this point $C$ is a cycle or a path}

if $C$ is a cycle then
    return the result of the c-planarity testing on $C$
else
    return true
end if
\end{verbatim}

Figure 6.9: Algorithm \textit{ClusteredCyclePlanarityTesting}
Second, we prove that algorithm ClusteredCyclePlanarityTesting can be always executed in $O(n^3)$ time, where $n$ is the number of vertices of $G$. In the first phase of the algorithm a cluster expansion is performed for all the clusters. By Property 17, each cluster expansion can be performed on a cluster $\mu$ in $O(n\mu)$ time, where $n\mu$ is the number of vertices of $\mu$. Therefore, this phase can be performed in linear time in the number of vertices of $G$. Also, at the end of this phase, by Property 16, the number of vertices is at most $2n$.

By Properties 16 and 19, each pair of cluster expansion and pipe contraction operations does not increase the number of vertices of the cycle. Also, by Properties 17 and 20, the two operations can be performed in linear time with respect to $n$.

Suppose that $E$ is the value of $E$ after the first phase of cluster expansions. We have that $E = O(n^2)$.

By Lemma 30 each pair of pipe contraction and cluster expansion operations decreases $E(C)$. Hence, the body of the while cycle is executed at most $E$ times. Also, contractible pipes can be determined in constant time using a suitable data structure that contains the candidate bases and that is updated after each operation. Therefore, algorithm ClusteredCyclePlanarityTesting works in $O(n^2) \times O(n)$ time, that is, $O(n^3)$ time.

6.5 Computing C-Planar Embeddings of Clustered Cycles

In this section we show how to build an embedding for a c-planar rigid clustered cycle. We assume that Algorithm ClusteredCyclePlanarityTesting, described in Section 6.4, has been applied, and that each step of the algorithm has been recorded. The clustered cycle $C_{end}$ obtained at the last step of the execution of the algorithm is such that $G^1(C_{end})$ is a cycle or a path. If $G^1(C_{end})$ is a cycle a c-planar embedding of $C_{end}$ can be easily computed by using the results described in Chapter 5. Otherwise, if $G^1(C_{end})$ is a path a c-planar embedding of $C_{end}$ can be computed by using the technique introduced in the proof of Lemma 31. The embedding of the input clustered cycle can be obtained by going through the transformations operated by Algorithm ClusteredCyclePlanarityTesting in reverse order starting from a c-planar embedding of $C_{end}$.

Algorithm ClusteredCyclePlanarityTesting performs two kind of operations: pipe contraction and cluster expansion. For each cluster expansion on a clustered cycle $C$, which produces a cluster cycle $C'$, the embedding of $C$ is directly obtained from the embedding of $C'$ since all pipes in $C'$ are also in $C$ and their embedding do not change (see the proof of Lemma 25).

For each pipe contraction on a clustered cycle $C$, which produces a cluster cycle $C'$,
6. Clustered Cycles

the embedding of $C$ can be computed starting from the embedding $\Lambda'$ of $C'$ as follows. Produce a drawing $\Gamma'$ of $C'$ according to $\Lambda'$. Produce a drawing $\Gamma$ of $C$ starting from $\Gamma'$ as described in the proof of Lemma 26. Extract the embedding $\Lambda$ from $\Gamma$.

Since obtaining a c-planar drawing from an embedding and vice versa can be performed in linear time, and since ClusteredCyclePlanarityTesting has a $O(n^3)$ time complexity, we can state the following result.

**Theorem 24** Given a c-planar rigid clustered cycle, a c-planar embedding of it can be computed in $O(n^3)$ time.
Chapter 7

An Example Application

In this chapter we show an example of application of clustered drawings in the context of network visualization. We propose a new visualization metaphor for the Internet Graph at AS level, which shows at the same time the AS graph and the Internet hierarchy.

7.1 The Visualization Problem

The Internet can be visualized at different abstraction levels for different purposes. For example, if one is interested in all the technical features of the network, then the Internet can be visualized with a high level of detail, showing the local area networks, the routers, and the point-to-point links. On the other extreme, if one is interested only in the relationships between Internet Service Providers (ISPs), then the Internet can be visualized at the Autonomous Systems (AS) level, where an AS is a collection of networks under the same administrative authority. At such a level, the Internet is currently partitioned into more than 20,000 ASes. An on-line survey about the visualization of the Internet can be found in [atl].

Many research efforts have been done for visualizing the Internet at the AS level. Some systems aim at visualizing the entire Internet structure (see, e.g., [HPMkc02, wal, CEH96, ine, ipm]), while others visualize a small portion of it with different specific purposes. For example [CDD+02] describes a system that visualizes the relationships between ASes that are stored into an Internet Registry. The BGPlay system [CDM+05] displays the evolution over time of the routing paths traversed by the traffic to a selected AS. BGPlay is a quite popular tool between ISPs operators and has been adopted by international organizations that publish real time collected data about
7. An Example Application

inter-AS routing [ris, ore]. A project that has a similar goal is presented in [MLZ05].

However, the research on the visualization of the Internet has to face new interesting challenges. In fact, recent studies [Gao01, DPP03] have shown that the ISPs (and then the ASes) of the Internet are self-organized into a customer-provider hierarchy (in the following Internet Hierarchy). In this hierarchy, customers buy from their providers a transit service for their traffic to and from the Internet. In [SARK02] each ISP is assigned to a level, according to its rank in the Internet Hierarchy. At this point, the problem of visually representing the Internet Hierarchy naturally arises.

An amazing number of methods have been devised to visually represent hierarchies in a variety of application domains (see, to give a few examples, [BFF05, YFDH01, BC03, BKW03]) many of which are targeted to show very large hierarchies.

In this chapter we address the problem of the simultaneous visualization of the Internet Hierarchy and of other Internet features that are interesting from some applicative perspective. More specifically, we extend the visualization paradigm that has been successfully adopted in BGPlay [CDM+05] enriching it with support for the Internet Hierarchy visualization. Another description of the new visualization paradigm can be found in [CDM+06]. Fig. 7.1 shows a screenshot of the BGPlay system. The red vertex (AS137), indicated by an arrow, is the AS we are focusing on. The picture represents paths to reach AS137 from several other ASes at a certain time. For example, the path AS16150, AS6939, AS6762, AS137 is shown to be used for traffic incoming AS137 from AS16150. No information is available in this visualization about the relevance of ASes and their economic relationships.

Fig. 7.2 shows a screenshot of BGPlay enhanced with the visualization system presented in this chapter. To visualize the Internet Hierarchy we use the metaphor of a topographic map. The contour lines are used to confine ASes that are at the same level of the hierarchy. For example, the ASes inside the central brown area are top level ASes. As the “hill” decreases in height the ASes decrease their rank in the hierarchy. The map shows quite well that some of the traffic flows have to climb the entire hierarchy to reach AS137, while other flows take “shortcuts”. For example, the path AS16150, AS6939, AS6762, AS137 does not pass through the Internet backbone since AS6939 and AS6762 exchange traffic at a lower level. Such kind of paths are usually more efficient and less expensive than paths that pass through the Internet backbone. An example of a real topographic map is show in Fig. 7.3; more detail about topographic maps can be found in [wik].

Observe that drawing a network within the above described metaphor can be interpreted as a special case of a clustered graph drawing problem, where the hierarchy describes a simple structure of clusters. Several authors dealt with the problem of representing a clustered graph using a spring embedder approach. Eades and Huang [EH00]
proposed a system for the visualization of huge graphs, by first performing a clustering and then visualizing a portion of the graph by applying a force-directed approach. Walshaw [Wal01] introduced an heuristic method for drawing large graphs which uses a multilevel technique combined with a force directed placement algorithm. Frishman and Tal [FT04] proposed an algorithm for dynamic drawings of clustered graphs.

### 7.2 Background

We provide some networking background and a brief description of the BGPlay routing visualization system.
Figure 7.2: A screenshot of the BGPlay system enhanced by the topographic map approach described in this chapter. The AS that originates the prefix is 137, highlighted in red and indicated by an arrow. Our approach clearly shows the importance of the ASes traversed by paths ending into 137.

Networking

In the Internet each host is identified by an IP address. An IP prefix identifies a set of (contiguous) IP addresses having the same leftmost $n$ bits, (e.g. 193.204.0.0/15 indicates a prefix 15 bits long) [Tan96]. Similarly to telephone call routing, inter-domain routing in the Internet is based on the destination prefix. Since a prefix identifies a set of addresses, it implicitly identifies a set of hosts having such addresses.

An Autonomous System (AS) is a portion of the Internet under a single adminis-
Figure 7.3: Examples of a real topographic map (courtesy of the U.S. GS).

trative authority. An Internet Service Provider (ISP) typically runs one or more ASes. Each AS is identified by an integer number. Traffic starting from an AS and directed to a specific prefix traverses a sequence of ASes called AS-path. The configuration of such paths on the routing devices is too complicated to be manually performed. Hence, ASes exchange routing information with other ASes by means of a routing protocol called Border Gateway Protocol (BGP) [Ste99]. BGP is based on a dis-
tributed architecture where border routers that belong to distinct ASes exchange the information they know about the reachability of prefixes. Two border routers that directly exchange information are said to have a a peering session between them, and the ASes they belong to are adjacent.

The AS-graph is the graph having one vertex for each AS and one edge between each pair of adjacent ASes. Note that the AS-graph is not a multi-graph.

Each router stores information about routing in its Routing Information Base (RIB). The RIB is a table where each row is a pair \( \langle \text{prefix}, \text{AS-path} \rangle \) meaning that a certain prefix is reachable through the associated AS-path. Such pairs are called routes. The main purpose of BGP is to allow the routers to exchange the routes they know. Since RIBs may be huge, the BGP process running on a router sends to its peers the full RIB only when a peering session is set up. During regular operation only changes to the RIB, called updates, are sent.

Routes related to a certain prefix begin their existence within an AS called the originator of the prefix (typically the AS to which the prefix belongs). These routes are propagated by means of updates to adjacent ASes, which in turn propagate it to adjacent ASes, etc. Every time a router propagates an update, it prepends its AS identifier to the AS-path; thus, the AS-path of an update is the list of ASes that the update has passed through.

The Internet Hierarchy

ASes cooperate in order to ensure good connectivity service to their customers but are competitors from a commercial point of view. Commercial relationships among ISPs directly influence how BGP routers are configured. Router configurations affect, in turn, the AS-paths that can be announced in the Internet. A valid AS-path is usually made of three consecutive parts [SARK02]: (i) it first traverses the Internet walking on the edges of the AS-graph in the direction from customers to providers, (ii) it optionally traverses one peer-to-peer relationship between two ASes of the same relevance, and (iii) it traverses the Internet walking on the edges of the AS-graph in the direction from providers to customers till the destination is reached.

In an ideal world, tier 1 ASes do not have any provider and are involved in peer-to-peer relationships with all the other tier 1 ASes; further, tier-i ASes are customers of ASes of tiers greater than \( i \), are involved in peer-to-peer relationships with other tier-i ASes, and are providers of ASes of tiers less than \( i \). From BGP routing tables it is possible to infer, with a few errors, the tier of the each AS [SARK02].
The BGPlay Internet Visualization Service

BGPlay is a service that displays the portion of the AS graph that describes how the traffic flows to a certain AS from a set of selected ASes. It obtains routing data from well-known and publicly available sources, namely the Routing Information Service (RIPE NCC) [ris] and the Route Views project (University of Oregon) [ore] whose archives are used for network debugging purposes or scientific investigation and are updated in real-time.

To query BGPlay, the user connects to a Web page, which hosts BGPlay, and starts the BGPlay applet which presents a query window asking for a prefix and a time interval. When the user submits the query, BGPlay processes the request and displays the animation window (Fig. 7.1), which presents the routing information.

The left part of the animation window contains the time panel, which plots the number of events over time on a logarithmic scale. The bottom of the panel corresponds to the start of the query interval and the top of the graph to the end; a small blue triangle indicates the current time (initially, the start of the query interval). The user may jump to a specific instant within the query interval by clicking on the time panel.

The main part of the window shows an AS-graph and the routing at a certain instant of time for the prefix specified by the user. Each number represents an AS, and the AS originating the prefix is placed in the center of the graph and highlighted by a red circle. Each solid or dashed line represents a segment of an AS-path seen by RIS or Oregon Route Views. An AS-path starts in the originating AS and stops in an AS which provides BGP routing data to RIS or Oregon Route Views. The AS-paths that do not change during the query interval are merged into trees rooted at the origin AS and drawn dashed. Each tree is drawn in different color so that it can be unambiguously identified. AS-paths that change during the query interval are drawn solid and are not merged.

Note that the graph may contain isolated nodes which have no paths to the origin AS. This does not necessarily imply that these ASes do not have a path to the queried prefix: more usually these ASes do not contain collector-peers (and thus no information about their routing is known) and appear in the graph because they have been or will be part of a path which was in use in another moment of the query interval.

BGPlay can animate the evolution of the routing over time. A panel, on the left of the window, shows the density of routing events over the interval of time of the query and highlights the instant of time whose routing is currently shown in the main part of the window.
7. **An Example Application**

7.3 **Choosing the Visualization Metaphor**

When we had to select a visualization metaphor for showing at the same time both the AS-paths of the BGP routing and the levels of the ASes hierarchy, we considered several possible options and focused on three main choices.

1. The classical approach of Sugiyama et al. [STT81] visualizes a hierarchy by assigning the y-coordinate of each vertex (AS) according to its rank. In this way, top (bottom) level vertices appear in the upper (lower) part of the drawing.

2. Another typically adopted strategy for visualizing a hierarchy [Car80] was to represent ASes with a size and/or color proportional to their rank in the hierarchy.

3. A further possibility was to extend the visualization approach already used for BGPlay [CDM+05] by placing the ASes within a “topographic map” where each AS falls into the region assigned to its level.

We deliberately ruled out three-dimensional representations, since, apart from the fascination of such a scenario, effectively browsing a three-dimensional environment involves the use of sophisticated interfaces, while network operators and researchers appear to be reluctant to adopt particularly complex software architectures and tools.

Before discarding options 1 and 2 we performed several experiments against typical routing scenarios obtained by querying the Routing Information Service of the RIPE NCC and equipping this information with the Internet hierarchy computed with the algorithm of [SARK02]. The purpose of the experiments was to compare the readability of the three graphic metaphors above from an information visualization perspective. To this aim we drew several diagrams according to the three strategies.

From our experiments we could observe the following pros and cons for the three metaphors

**The traditional hierarchical representation** through layered drawings, although very effective in conveying the hierarchy information associated with the ASes, turned out to be unsatisfactory for our application. In layered drawings, in fact, the x-coordinates of the nodes are generally chosen in order to reduce edge crossings. It is often the case that a climbing path zigzags horizontally the diagram (see Fig. 7.4), reducing its readability. Also, two nodes which are one near the other on the same level (see the black and the gray nodes of Fig. 7.4) may represent ASes that are actually very distant in the network. On the other hand, the number of edge crossings jumps up when the x-coordinates of the nodes are somehow constrained. Finally, many of the algorithms used for producing hierarchical representations do not allow
Choosing the Visualization Metaphor

Figure 7.4: An handmade layered drawing showing the drawbacks of this kind of representation. Observe how paths zigzag while climbing and descending the hierarchy. Also, ASes that are very far in the network may be represented by nodes which are very near in the drawing (as the black and the gray nodes).

Using size and color to suggest the rank of the AS in the hierarchy has the advantage of being a very simple strategy, easy to implement and to plug into existing visualization systems, but, on the other hand has the major disadvantage of depriving the user of an overall view on the ASes of the same level. ASes are scattered in the drawing, and there is no clue of how many levels of the hierarchy are involved and how many links span more than one level.

The “topographic map” strategy produced the most clear and intuitive drawings. In such metaphor, ASes of the same rank are drawn inside the same region and regions corresponding to a higher level are contained inside the regions corresponding to the lower ones. Further, this metaphor is consistent with well-known drawing conventions in the area of inter-domain routing where the AS-paths are merged into a graph, which is colored in such a way to make it possible to uniquely reconstruct the path from each AS to the AS advertising the prefix.

More in detail, in our approach, the tiers of the ASes play the role of the elevation. We can imagine the drawing as a topographic map in which ASes of tier 1, the top level, are drawn in the middle, resembling the peak of a mountain, and ASes of lower levels are drawn progressively farther, as if they were on the downhill of the mountain.
More formally, each tier \( i \) is associated with a connected region of the plane that we denote \( A_i \) with \( i = 1 \ldots T \), where \( T \) is the number of tiers. Region \( A_i \), with \( i = 2, \ldots, T \), has a hole which contains region \( A_{i-1} \). All and only ASes of tier \( i \) are contained into region \( A_i \). The shape of the regions is a critical issue, since it constrains the layout of the ASes and hence may reduce the effectiveness of a layout algorithm in obtaining drawings with small edge length, even ASes distribution, and small number of crossings. In an ideal topographic map the regions have not necessarily a regular shape, but smoothly surround the ASes that they contain. Region shapes and ASes positions are strictly related, since in order to be enclosed into a region the ASes of the same tier should be near one to the other.

We recall that the edge region crossings of a map is the number of edges that cross the border of the same region more than once. Consider that when an edge connects two nodes in different regions a crossing between such edge and a region border occurs. On the other hand, each edge should cross the border of a region at most once. For readability reasons, we prefer topographic maps with no edge region crossing.

The algorithm described in Section 7.4 produces such drawings which look very much like a topographic map with a superimposed AS-graph.

Concerning the representation of the AS-paths and of the way they change over time, we adopt the same drawing standard adopted by BGPlay and described in Section 7.2. An example of the final result is shown in Fig. 7.2.

7.4 Layout Algorithm

In this section we describe the layout algorithm for producing drawings according to the visualization metaphor described in Section 7.3.

We adopt a force directed approach, also called spring embedder, which computes the final layout of the input graph as the equilibrium configuration of a simulated physical system where the nodes are represented by mutually repulsing electric charges and the edges by springs with a given stiffness and a natural length [FR91, DETT99]. Several factors encourage such a choice: (i) Since its physical system can be enriched with additional forces, a spring embedder is easily tailorable to account for additional constraints on the drawings [SM95a, SM95b, DETT99]. (ii) Spring embedders have already been used to produce drawings of clustered graphs in which the nodes of each cluster are constrained to stay inside the same region of the plane [EH00]. (iii) The original BGPlay interface obtains pleasant and effective results with a spring embedder algorithm.

Our visualization metaphor is quite different from the drawing conventions for
which spring embedders are usually applied. In fact, our regions are all contained one inside the other and their boundaries have not a fixed shape. Therefore, we devised a new model whose main ingredients are the following.

- In order to enclose one region into the other, and force the ASes to belong to their assigned region, we introduce one border region $B_i$ at a time, starting from the inner region $B_1$ and ending with $B_{T-1}$, where $T$ is the number of tiers (ASes of tier $T$ have no boundary). When a border is introduced, additional forces are added to the physical model to ensure that the ASes are enclosed into the border of their region. In this phase the region borders are constrained to be circular. The incremental introduction of the borders will make edge region crossings unlikely to occur.

- In order to allow borders $B_i$ to change their shape and adapt to the position of the ASes (as required by the topographic metaphor described in Section 7.3), we represent each border with a chain of very close special nodes, called *border nodes*. When all region borders are introduced, these border nodes are allowed to move. Also, by suitably removing or adding border nodes to the chains, we control their density, since a loose chain of border nodes would easily be traversed by an AS, while a too dense chain would wind on itself.

A detailed description of our layout algorithm follows. The input of the algorithm is the AS-graph where each AS is labeled with a tier number in $\{1, \ldots, T\}$. The output of the algorithm is a layout of the AS-graph and the borders that separate areas $A_1, \ldots, A_T$ where each area contains all ASes associated with its tier as stated in Section 7.3.

The layout algorithm is composed of two phases.

**Phase 1.** A set of $T - 1$ concentric circular borders $B_1, \ldots, B_{T-1}$ are identified on the plane, with radius $r_1 < r_2 < \ldots < r_{T-1}$. We denote by $A_1$ the area enclosed by $B_1$ and by $A_i$, with $i = 2, \ldots, T - 1$, the area enclosed by $B_i$ and outside $B_{i-1}$. The purpose of this phase is to ensure that, for each tier $i$, all ASes belonging to tier $i$ are placed inside area $A_i$. This is obtained by means of an incremental layout strategy that starts from ASes of Tier 1 (see Figs. 7.5, 7.6, 7.7).

Consider, for each boundary $B_i$ of radius $r_i$, circles $B_i^-$ and $B_i^+$ with radius $r_i^- = r_i - \epsilon$ and $r_i^+ = r_i + \epsilon$ respectively, where $\epsilon$ is a very small number with respect to the radius of $B_i$.

1. ASes of Tier 1 are confined inside $B_1^-$ by a force $f_1^-$ that attracts these ASes toward the center of the drawing and that is null outside $B_1^-$. All other ASes
Figure 7.5: Example of application of the algorithm. Phase 1, the addition of the first fence.

are kept outside $B_1^{+}$ by a force $f_1^{+}$ that repulses these ASes from the center of the drawing and that is null inside $B_1^{+}$. With these additional forces the spring embedder is run until an equilibrium is reached.

2. The space between $B_{i}^{-}$ and $B_{i}^{+}$ is used to place a chain of border nodes called fence. Border nodes cannot move in this phase but they repel ASes and curb them inside their region. With $B_1$ represented by border nodes the spring embedder is run until equilibrium is reached. See Fig. 7.5.

3. Operations of steps 1 and 2 are iteratively applied for Tiers $2, \ldots, T - 1$. That is, for each Tier $i$, the spring embedder is run with new forces, $f_i^{-}$ and $f_i^{+}$, to contain inside $B_i^{-}$ vertices of tiers less than or equal to $i$ and outside $B_i^{+}$ vertices of tiers greater than $i$. A new fence is added and a new spring embedder run is performed. Throughout the computation, the forces introduced in the previous steps are not dropped, but remain to enforce proper confinement of each AS within its area. See Figs. 7.6, 7.7.
Figure 7.6: Example of application of the algorithm. Phase 1, the addition of the second fence.

**Phase 2.** Forces $f_i^-$ and $f_i^+$, for $i = 1, \ldots, T - 1$, are removed, and the border nodes are left free to move until equilibrium is reached. In order to keep the border density within a safe range, edges of the fences that are longer than a certain threshold are split and edges that are shorter than a certain threshold are contracted. Also, the force exerted on a border node is multiplied by a *mobility factor* that allows to fine-tune the rigidity of the fences. An example of the result of Phase 2 is shown in Fig. 7.8, while the corresponding picture presented to the user is shown in Fig. 7.2.

The algorithm described above has been prototypically implemented in the BG-Play system. No change has been made to the BGPlay architecture described in [CDM05]. The information about levels of the ASes is computed with the techniques shown in [SARK02]. Our experiments have been performed on a PC with an AMD AthlonXP 2600 processor – 2.133 GHz and with 0.5 GB of RAM. For our test suite running times range from 7 to 62 seconds with an average 27.83 seconds.

In the following we provide a more detailed description of the layout algorithm; the description includes the specification of the actual parameter of the algorithm.

1. each AS is placed in (0; 0) and the length at rest of each edge is set to 60.
7. An Example Application

Figure 7.7: Example of application of the algorithm. Phase 1, the addition of the fourth fence.

2. initialize $C'_1$ and $C''_1$ such that $r'_1 = r_1 - \varepsilon$ and $r''_1 = r_1 + \varepsilon$.

3. initialize $C_{ext}$ with center in $(0; 0)$ and $r_{ext}$ such that $C_{ext}$ contains the whole drawing.

4. run the standard spring embedder algorithm plus the following forces:
   - a force attracting the Tier 1 ASes toward $(0; 0)$; this force is null outside $C'_1$.
   - a force repulsing the other ASes toward the external of the drawing; this force is null inside $C''_1$.
   - a force attracting all the ASes toward $(0; 0)$; this force is null outside $C_{ext}$. This last force will be present in each application of the spring embedder.

A the end of this step the Tier 1 ASes are placed near $(0; 0)$, while the other ASes are in generic positions around these ASes.

5. create the fence for the Tier 1 ASes. The number of nodes in this fence is proportional to the number of Tier 1 ASes. these nodes are positioned along $C_1$. 

Figure 7.8: Example of application of the algorithm. Final layout after Phase 2. The actual picture shown to the user is presented in Fig. 7.2.

6. run the standard spring embedder algorithm on the AS graph and the first fence, with the same set of forces of the previous application. This second application of the spring embedder produces a better distribution of the ASes which are outside $C_1$.

7. initialize $C'_2$ and $C''_2$ such that $r'_2 = r_2 - \varepsilon$ and $r''_2 = r_2 + \varepsilon$.

8. run a standard spring embedder plus the same forces described in step’4 plus new forces related to $C'_2$ and $C''_2$.

9. create the fence for Tier 2 ASes.

10. set the length at rest for the edges of the AS graph to 40

11. repeat steps 8 and 9 for the Tier 3 and tier-4 ASes,

12. set the length at rest of the edges to 10

13. set the minimum and maximum length for the edges of the fences.
7. An Example Application

14. unfreeze the fences and run a standard spring embedder. In this step it is possible to split or contract the edges of the fences, according to the edges’ length.

15. fill the areas $A_1, \ldots, A_T$ with the corresponding colors.

7.5 Experimental Evaluation

In order to experimentally evaluate the effectiveness of the proposed algorithm we need to consider both traditional graph drawing aesthetic criteria and aesthetic criteria suitable for the specific application and the adopted visualization metaphor.

Concerning the first ones, in the graph drawing and visualization literature several measures have been proposed to evaluate the aesthetic quality of the straight-line drawings obtained with force directed methods (see, e.g., [BHR96]). We decided to adopt the following measures. **Normalized number of edge crossings** (EEX), i.e. the ratio between the number of edge crossings and the total number of edges, which is generally considered the most prominent readability measure [Pur00, Pur02]. **Normalized edge-length standard deviation**, i.e., the ratio between the standard deviation and the average length of the edges (ELSD). In fact, a readable drawing should have an homogeneous length for its edges. **Node distance ratio** (NDR), i.e., the ratio between the distance of the nearest pair of nodes and that of the farthest one. This measures the “resolution rule” of the drawing [DETT99].

Concerning aesthetic criteria that are specific for the application, we selected the following ones. **Edge region crossings** (ERX), i.e., the number of edges that cross the border of the same region more than once. **Coastline indentation vector** (CIV), that is an index $CI_i$ for each region $i$ measuring how smooth is the border of that region. In fact, we noticed that “nice looking” regions have borders which are not too much indented. We define $CI_i = \frac{b_i}{2\sqrt{a_i}}$, where $b_i$ and $a_i$ are the outer perimeter of border $B_i$, and the area enclosed by it, respectively. Observe that, $CI_i \geq 1$, where $CI_i = 1$ if $B_i$ is drawn as a circle. Also, it would be reasonable to have a coastline indentation that slightly increases with $i$, thus having $CI_i \leq CI_{i+1}$. **Density variation vector** (DVV), that measures how the density changes from region to region. The density variation between subsequent areas is defined as $DV_i = \frac{D_i - D_{i+1}}{D_i}$, where $D_i$ is the number of nodes in region $i$ divided by $A_i$. Essentially, we would like to have drawings that show a small change of their density when moving from a region to an adjacent one.

The number of prefixes currently announced in the Internet is about 200,000 originated by about 20,000 ASes. To perform our experiments we selected a test suite of 141 prefixes in the following way. We selected all the Tier 1 ASes (which are 21). Then, we randomly selected 30 ASes in each tier. For each selected AS we randomly picked up a prefix. For each prefix we applied the algorithm described in Section 7.4
with four different choices of the rigidity of the fences. We shall refer to such different versions of the algorithm with CIRCLES, RIGID FENCES, SOFT FENCES, and LOOSE FENCES. CIRCLES corresponds to having fixed fences, with the shape of circles, which is equivalent to set the mobility factor to zero (see Section 7.4). It constitutes a reference point for the quality of the drawings. SOFT FENCES corresponds to the choice we have done in the current version of the system. LOOSE FENCES have the mobility factor equal to 4/3 the mobility factor used by SOFT FENCES. Finally, RIGID FENCES corresponds to an intermediate choice between CIRCLES and SOFT FENCES and has a mobility factor equal to 1/3 of SOFT FENCES.

All drawings computed with all four versions of the algorithm have ERX=0.

Fig. 7.9 shows the density of the values of EEX measured in our experiments. Namely, the x-axis reports the values of EEX while the y-axis reports the number of drawings having that value of EEX. Since EEX is a continuous variable, we grouped its values in 15 intervals. For example, the y-value 11 corresponding to the x-value 0.605 on the curve for CIRCLES means that there are 11 drawings with value of EEX between 0.56 and 0.65. We use density charts also for the other quality measures.

As Fig. 7.9 shows, SOFT FENCES gives values of EEX that are lower, in average, with respect to CIRCLES. A similar behavior, but less marked, we have for RIGID FENCES. LOOSE FENCES shows only a marginal improvement on SOFT FENCES. The average EEX values are 0.189 for LOOSE FENCES, 0.196 for SOFT FENCES, 0.364 for CIRCLES, and 0.24 for RIGID FENCES. This happens because fixed or rigid fences do not allow (or only partially allow) the spring embedder algorithm to disentangle the drawing.

Fig. 7.10 and 7.11 shows the density of the values of NDR and ELSD, respectively. Both figures put in evidence the beneficial effect of having fences that are free to move. Again, SOFT FENCES and LOOSE FENCES show a very similar behavior.

Figs. 7.12 and 7.13 show the density of CI_1 and CI_2. The plots for CI_1 and CI_4 are similar to that of CI_2. The plots put in evidence that the satisfactory values of EEX, NDR, and ELSD are obtained at the expense of larger contour indentation. The worst cases are the ones of CI_1 and CI_2. This is somehow counter-intuitive since we could expect higher indentation for the most external regions. However, such regions have typically less nodes. Hence, the contours tend to stretch. Also, for these measures, it is apparent that LOOSE FENCES performs worse than SOFT FENCES.

Fig. 7.14 shows that SOFT FENCES and LOOSE FENCES perform much better than the other settings in terms of the density variation between subsequent areas. This is another positive effect of having fences that can move. For brevity we omit results for DV_2 and DV_3 that are quite similar.

Overall, we think that SOFT FENCES is a good trade-off between the quality of the borders and the quality of the graph layout. LOOSE FENCES offers a small improve-
7. AN EXAMPLE APPLICATION

Figure 7.9: Density of EEX (normalized number of edge crossings). CIRCLES and RIGID FENCES are more affected by edge crossings.

We think that the limitations of our approach are basically the same of conventional spring embedders, which are known to perform poorly for graphs with high density (ratio between the number of edges and the number of nodes). However, in our application, the density is quite low. In our experiments density ranges from 0.98 to 1.16 (average is 1.04). The drawing of the AS-graph with the highest density is shown in Fig. 7.15.

Three network topographic maps of the same AS-graph produced for the evaluation are in Fig. 7.16.

We have done some evaluating interviews with ISP network managers to get subjective evaluative remarks. Namely, we proposed our prototype to people working with NaMeX [nam], the second Internet Exchange Point in Italy, grouping 22 large and medium size ISPs. We asked for comments and feedbacks on the usability of the
Experimental Evaluation

Figure 7.10: Density of NDR (node distance ratio). LOOSE FENCES and SOFT FENCES show a better “resolution rule” for their nodes.

...system, also compared with the current version of BGPlay. They found interesting and useful to have the relevance of each AS highlighted. The choice of the topographic approach has been appreciated, as well as the quality of the drawings. We collected feedback on minor usability issues related to interaction that were easy to amend...
Figure 7.11: Density of ELSD (normalized edge-length standard deviation). LOOSE FENCES and SOFT FENCES show more homogeneous edge lengths.
Figure 7.12: Density of $CI_1$ (coastline indention of region 1). Augmenting the mobility factor progressively increases the indention of the border of region 1.
7. An Example Application

Figure 7.13: Density of $C_{I2}$ (coastline indentation of region 2). LOOSE FENCES and SOFT FENCES have a comparable quality with respect of this measure. CIRCLES and RIGID FENCES produce, obviously, smoother borders.
Figure 7.14: Density of $DV_1$ (density variation between areas $A_1$ and $A_2$). A more homogeneous density is shown by LOOSE FENCES and SOFT FENCES.
7. An Example Application

Figure 7.15: The map of the AS-graph with highest density in the test suite (Algorithm SOFT FENCES).

Figure 7.16: Three topographic maps of the same AS-graph. They were produced by different algorithms: (a) CIRCLES, (b) RIGID FENCES, and (c) SOFT FENCES.
Conclusions

This thesis offers several contributions in the field of c-planarity testing. With respect to a c-connected clustered graph $C(G, T)$, we proposed a structural characterization of c-planarity, both in the case that the underlying graph $G$ is biconnected and in the general case. Based on such a characterization, we provided a linear time algorithm to test the c-planarity of $C$. If $C$ is non-c-planar, our algorithm identifies a structural element responsible for non-c-planarity. The algorithm is fully described in terms of elementary steps and it is easily implementable in linear time.

Furthermore, we dealt with the problem of c-planarity testing in the general case of non-connected graphs. We defined several classes of highly non-connected clustered graphs, introducing new test methodologies and polynomial time c-planarity algorithms.

We also proposed an example of application of clustered drawing in the context of network visualization.

Of course, several problems remain open. The main problem of the c-planarity field is certainly to state the computational complexity of the c-planarity testing for general (non-connected) clustered graphs. However, the c-planarity of c-connected clustered graphs still deserves further investigation. For example, it would be really interesting to provide in this context a characterization in terms of obstructive patterns, analogous to $K_{3,3}$ and $K_5$ in the Kuratowski’s theorem. Also, the investigation of c-planarity with additional constraints may be interesting from an applicative point of view. The problem of the general planarity testing is interesting, in our opinion, even if restricted to simpler sub-cases.

- Are there other families of unconnected clustered graphs whose c-planarity can be efficiently assessed and whose underlying graph has a simple structure? For example, what happens if the underlying graph is a tree or a series-parallel graph? It is easy to show that a flat clustered graph whose underlying graph is a path and such that graph of the clusters $G_1$ is a cycle, is c-planar. It is also easy
to find an example of an unconnected flat clustered graph whose underlying graph is a tree, such that $G^1$ is a cycle and that is not $c$-planar (see Fig. 1).

- Suppose that the underlying graph has a fixed embedding. Can this hypothesis simplify the $c$-planarity testing?

- Can the techniques introduced in Chapters 5 and 6 be combined with techniques known in the literature for $c$-planarity testing and embedding problem for more complex families of clustered graphs? Observe that a trivial generalization of the result to flat clustered graphs whose underlying graph is a general graph fails. In fact, it is easily to find clustered graphs which are not $c$-planar while all cycles of their underlying graphs are separately $c$-planar (see Fig. 2).

![Figure 1: A 3-cluster graph that is not $c$-planar. The underlying graph is a tree.](image)

With respect to the visualization problem proposed in Chapter 7, as future lines of research we suggest the following:

- combine the current information about Internet routing with information on the amount of traffic flows from and to the focal ISP

- integrate the inter-domain routing information currently provided by BGPlay with intra-domain routing

- compute AS hierarchies with algorithms alternative to [SARK02].
Figure 2: An example that shows that the c-planarity for cycles is only a necessary condition but not a sufficient one for the c-planarity of more complex graphs. The graph of the clusters is supposed to have fixed embedding while the underlying graph is planar and composed by three paths between two vertices (each path is drawn in the picture with a different line-style).
Bibliography

[atl] The atlas of cyberspace.
http://www.cybergeography.org/atlas/.


[GLS05] M. T. Goodrich, G. S. Lueker, and J. Z. Sun. C-planarity of extro- 
vert clustered graphs. In P. Healy and N.S. Nikolov, editors, Proc. 
Graph Drawing 2005 (GD’05), volume 3843 of LNCS, pages 211–222. 
Springer-Verlag, 2005.

In Joe Marks, editor, Proc. Graph Drawing 2000 (GD’00), pages 77–90. 


[HPMkc02] B. Huffaker, D. Plummer, D. Moore, and k claffy. Topology discovery by 
active probing. In Symposium on Applications and the Internet (SAINT), 
2002. 

[Hsu03] W.L. Hsu. An efficient implementation fo the PC-Tree algorithm of Shih 

1974.

[HU79] J. E. Hopcroft and J. D. Ullman. Introduction to Automata Theory, Lan-

[ine] The internet mapping project. 


[JM96] M. Jünger and P. Mutzel. Maximum planar subgraphs and nice embed-
issue on Graph Drawing, edited by G. Di Battista and R. Tamassia).


[LEC67] A. Lempel, S. Even, and I. Cederbaum. An algorithm for planarity test-


