# Clustered Level Planarity*

Michael Forster and Christian Bachmaier

University of Passau, 94030 Passau, Germany
{forster,bachmaier}@fmi.uni-passau.de

**Abstract.** Planarity is an important concept in graph drawing. It is generally accepted that planar drawings are well understandable. Recently, several variations of planarity have been studied for advanced graph concepts such as $k$-level graphs [6,10–16] and clustered graphs [5,7]. In $k$-level graphs, the vertices are partitioned into $k$ levels and the vertices of one level are drawn on a horizontal line. In clustered graphs, there is a recursive clustering of the vertices according to a given nesting relation. In this paper we combine the concepts of level planarity and clustering and introduce clustered $k$-level graphs. For connected clustered level graphs we show that clustered $k$-level planarity can be tested in $\mathcal{O}(k|V|)$ time.

## 1 Introduction

Many structures in real life applications cannot be represented appropriately by standard graphs. In biochemical pathways, for example, substances and reactions can be modelled as vertices and (hyper) edges. But often it is also desirable to visualise the cell compartments containing distinct substances and reactions. These compartments define different components (clusters) of the graph and are recursively nested. This leads to an extended graph model: A *clustered graph* $G = (V, E, \Gamma)$ is a directed or undirected graph with an additional recursive nesting relation $\Gamma = (V_\Gamma, E_\Gamma)$ with $V_\Gamma = C \cup V$. $\Gamma$ is a rooted tree of the clusters $C$ as inner nodes and the leaves $V$. Each cluster $c \in C$ represents a subgraph $G_c = (V_c, E_c)$ of $G$ induced by its descendant leaves $V_c$. It can be assumed that each cluster has at least two children. Thus the number of clusters $|C|$ is linear. Clustered graphs are drawn such that the clusters are simple closed curves that define closed regions of the plane. The region of each cluster contains exactly the clustered drawing of the subgraph induced by its vertices. Regions are nested recursively according to $\Gamma$. A clustered graph is *c-planar* if it has a drawing without edge crossings, region intersections, or crossings between an edge and a region. An edge crosses a region if it crosses its border at least twice. It is an open problem, see e. g. [5,7,9], whether c-planarity can be tested efficiently. Here connectivity plays a crucial role. It can be done in linear time if the clustered graph is *c-connected*, i. e., if each subgraph induced by a cluster is connected, see [5,7].

* This research has been supported in part by the Deutsche Forschungsgemeinschaft, grant BR 835/9-1.

For drawing clustered graphs there are different approaches. See [2] for an overview. If the underlying graph is directed, then Sander's extension of the Sugiyama algorithm [17–19] can be used. The first step of this algorithm is to distribute the vertices to $k$ levels. Then it uses heuristics to reduce the number of edge crossings, see also [8], because the minimisation problem is NP-hard. It does not even guarantee a planar drawing if one is possible. Since in this case it is especially desirable to avoid crossings we analyse this problem called clustered level planarity. This is the combination of c-planarity and level planarity.

A *$k$-level graph* $G = (V, E, \phi)$ is a directed or undirected graph with a level assignment $\phi\colon V \to \{1, 2, \ldots, k\}$, $1 \leq k \leq |V|$, that partitions the vertex set into $V = V^1 \dot\cup V^2 \dot\cup \cdots \dot\cup V^k$, $V^i = \phi^{-1}(i)$, $1 \leq i \leq k$, such that $\phi(u) \neq \phi(v)$ for each edge $(u, v) \in E$. A $k$-level graph $G$ is $k$-level planar if it is possible to draw it in the Cartesian plane such that all vertices $v \in V^i$ of the $i$-th level are placed on a single horizontal line $l_i = \{\, (x, i) \mid x \in \mathbb{R} \,\}$ and the edges are drawn as vertically strictly monotone curves without crossings. A *clustered $k$-level graph* $G = (V, E, \Gamma, \phi)$, or short *cl-graph*, is a $k$-level graph with a recursive clustering. For every cluster $c \in C$ denote the minimum and maximum levels with vertices in $c$ by $\phi_{\min}(c)$ and $\phi_{\max}(c)$. A clustered $k$-level graph $G$ is *clustered $k$-level planar* or short *cl-planar* if it has a level planar embedding in the plane such that the following restrictions are satisfied:

R1 Each intersection of a level line and a cluster region is a single interval, i. e., the vertices on each level that belong to the same cluster are placed next to each other with no other vertices in between.
R2 Clusters do not cross each other, i. e., the relative position of two clusters is the same on all levels.
R3 Edges do not cross clusters, i. e., no edge intersects the border of a cluster region twice.

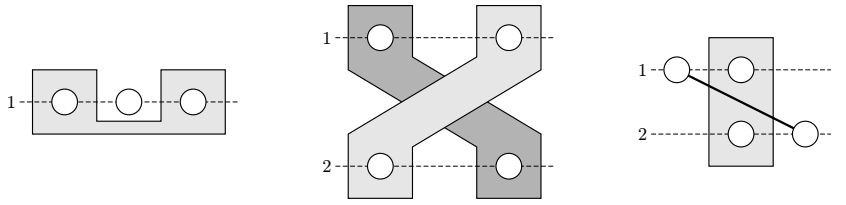
See Fig. 1 for a visualisation of R1–R3. These restrictions ensure that any cl-planar graph can be drawn without crossings such that all cluster regions are convex. They can even be drawn as rectangles by using Sander's algorithm.

**Lemma 1.** *If $G = (V, E, \Gamma, \phi)$ is a clustered $k$-level graph, then obviously*

1. *$G$ is cl-planar $\Rightarrow$ $(V, E, \phi)$ is level planar $\Rightarrow$ $(V, E)$ is planar.*
2. *$G$ is cl-planar $\Rightarrow$ $(V, E, \Gamma)$ is c-planar $\Rightarrow$ $(V, E)$ is planar.*

Note that a level planar and c-planar cl-graph is not necessarily cl-planar. Figure 2 is a counter-example. Without loss of generality we consider only simple graphs without self loops and parallel edges. Because of Lemma 1 a simple input graph with $|E| > 3|V| - 6$ is rejected as not cl-planar and we can assume that the number of edges is linear in the number of vertices.

We give an $\mathcal{O}(k|V|)$ time algorithm based on the previous work of Di Battista and Nardelli [6]. It solves cl-planarity for cl-graphs that are proper, level connected, and hierarchies: A *hierarchy* is a level graph with a single *source* and an arbitrary number of *sinks*, where a source is a vertex having only edges to a




(a) Violation of R1 (b) Violation of R2 (c) Violation of R3

**Fig. 1.** Violations of the cl-planarity restrictions

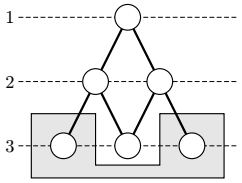



**Fig. 2.** A cl-graph that is level planar and c-planar but not cl-planar

higher level and a sink is defined analogously. Note that hierarchies should not be confused with the hierarchical clustering $\Gamma$. A (clustered) $k$-level graph is *proper* if $|\phi(u) - \phi(v)| = 1$ for each edge $(u, v) \in E$. A clustered level graph is *level connected* if any two consecutive levels of the same cluster are spanned by an edge of the cluster, i. e., if $\forall c \in C\colon \forall i \in \{\phi_{\min}(c), \ldots, \phi_{\max}(c) - 1\}\colon \exists (u, v) \in E_c\colon \phi(u) \leq i \wedge \phi(v) \geq i + 1$. Level connectivity follows directly from c-connectivity.

## 2 Related Work

There are several algorithms for level planarity testing. The first is from Di Battista and Nardelli [6]. It is restricted to hierarchies and runs in linear time. Chandramouli and Diwan [3] present a linear time algorithm for triconnected DAGs. Leipert et al. [13–16], based on the work of Heath and Pemmaraju [11,12], finally come up with a linear time algorithm for general level graphs. Their algorithm is based on [6] and it is also able to compute a level planar embedding. Since this algorithm is rather complex there is a simpler approach of Healy and Kusik which runs in $\mathcal{O}(|V|^2)$ time on proper graphs and computes an embedding in $\mathcal{O}(|V|^3)$ time. Since our algorithm extends the algorithm of [6] we have to recall some details which are necessary to understand our extensions. We use a simplified notation.

The basic idea of the algorithm is to perform a top down sweep, processing the levels in order $1, 2, \ldots, k$ and for every level $V^i$ compute a set of permutations of

$V^i$ that appear in some level planar embedding of $G^i$. $G^i$ is the subgraph induced by $V^1 \cup V^2 \cup \cdots \cup V^i$. $G$ is level planar if and only if the set of permutations of $G^k = G$ is not empty.

In order to represent and store sets of vertex permutations efficiently, a data structure called PQ-tree, introduced by Booth and Lueker [1], is used. A PQ-tree represents the permutations of a finite set $S$ in which the members of specified subsets of $S$ occur consecutively. It is a rooted and ordered tree with leaves and two types of inner nodes, P- and Q-nodes. In the context of this paper the term vertex denotes an element of a graph and the term node denotes an element of a PQ-tree or of a cluster tree. The leaves correspond to the elements of $S$ and the possible permutations are encoded by the combination of the two types of inner nodes. The children of a P-node can be permuted arbitrarily, whereas the children of a Q-node are ordered and this order can only be reversed. If PQ-trees are used in planarity tests, a P-node always represents a cut vertex and a Q-node represents a biconnected component of a graph. The leaves represent edges to the not yet processed part of the graph. If there are no permutations with the given restrictions, the PQ-tree is empty. The most important operation on PQ-trees is REDUCE. It restricts the encoded set of permutations such that all elements of a given set $S' \subseteq S$ are consecutive in all remaining permutations. During the reduce phase, PQ-leaves representing elements of $S'$ are called *pertinent*. The minimum subtree containing all pertinent leaves is called the *pertinent subtree*. All its nodes are said to be pertinent, too. Its root is called *pert-root*.

Algorithm 1 describes the LEVEL-PLANARITY-TEST for hierarchies where $T(G^i)$ is a PQ-tree representing the set of admissible permutations of the vertices in $V^i$ that appear in some level planar embedding of $G^i$. The procedure CHECK-LEVEL realises a sweep over a single level $i$. All operations are performed in $T(G^i)$. Define $H^i$ to be the *extended form* of $G^i$. It consists of $G^i$ plus some new *virtual vertices* and *virtual edges*. For every edge $(u, v)$ with $u \in V^i$ and $v \in V^{i+1}$, a new virtual vertex with label $v$ and a virtual edge connecting it to $u$ is introduced in $H^i$. Note that there may be several virtual vertices with the same label but each with exactly one entering edge. The extension of $T(G^i)$ to $T(H^i)$ is accomplished by a PQ-tree operation called REPLACE_PERT. Since all PQ-leaves with the same label appear consecutively after the PQ-tree oper-

**Algorithm 1:** LEVEL-PLANARITY-TEST

**Input**: A level graph $G = (V^1 \dot\cup V^2 \dot\cup \ldots \dot\cup V^k, E, \phi)$

**Output**: boolean value indicating whether $G$ is level planar

Initialise $T(G^1)$

**for** $i \leftarrow 1$ ***to*** $k - 1$ **do**
  $T(G^{i+1}) \leftarrow$ CHECK-LEVEL($T(G^i)$, $V^{i+1}$)
  **if** $T(G^{i+1}) = \emptyset$ **then return false**
**end**

**return true**

ation REDUCE in every admissible permutation, each consecutive set of PQ-leaves is replaced with REPLACE_PERT by a P-node. This is the parent of new leaves representing the adjacent vertices of $v$ in $V^{i+1}$. Afterwards all PQ-leaves representing vertices in $V^{i+1}$ with the same label are reduced to appear as a consecutive sequence in any permutation stored in the PQ-tree. $R^i$ denotes this *reduced extended form* of $H^i$. Finally in a new sweep over this level all PQ-leaves representing sinks $v$ in $V^{i+1}$ are deleted from the PQ-tree reconstructing it such that it obeys the properties of a valid PQ-tree again. See [1,11–16] for details on these operations.

## 3 Algorithm

### 3.1 Idea

Our algorithm for testing cl-planarity of proper and level connected hierarchies is based on the PQ-tree method for level planarity testing of hierarchies of [6]. This method already ensures that the calculated embedding is level planar. It remains to show how the additional properties R1–R3 for cl-planar embeddings can be maintained. We will see later that R2 and R3 are automatically satisfied if the graph is level connected, while an extension is necessary for R1.

An analysis of R1 reveals a similarity to the ordering constraints of PQ-trees because the vertices of one cluster have to be placed next to each other. This corresponds directly to the semantics of the REDUCE operation which restricts the set of admissible permutations to those where the PQ-leaves given as an argument appear consecutively. We obtain the following idea: The level by level sweep of the level planarity testing algorithm remains the same. The admissible permutations are stored in a PQ-tree $T$. We ensure R1 by additional applications of REDUCE. This is done by an extension of CHECK-LEVEL, see Algorithm 2. On each level a new method REDUCE-CLUSTERS is called, which ensures that the interior of each cluster is consecutive.

**Algorithm 2:** CHECK-LEVEL

**Input**: PQ-tree $T(G^i)$ of the current level, Vertices $V^{i+1}$ of the next level

**Output**: PQ-tree $T(G^{i+1})$ of the next level

extend $T(G^i)$ to $T(H^i)$
reduce $T(H^i)$ to $T(R^i)$
**if** $T(R^i) = \emptyset$ **then return** $T(G^{i+1}) \leftarrow \emptyset$
REDUCE-CLUSTERS($T(R^i)$, $V^{i+1}$) *// NEW*
remove sinks from $T(R^i)$
**return** $T(G^{i+1}) \leftarrow T(R^i)$

### 3.2 Efficient Cluster Reduction

A straightforward implementation of the REDUCE-CLUSTERS method is to call REDUCE for the PQ-leaves of each cluster. This leads to a running time of $\mathcal{O}(k|C||V|)$, i. e., up to $\mathcal{O}(k|V|^2)$, since for each of the $k$ levels and for each of the |C| clusters the whole PQ-tree of size $\mathcal{O}(|V|)$ must be traversed. With the following approach this can be improved to $\mathcal{O}(k|V|)$ time.

First consider only two clusters $c_1$ and $c_2$ on the same level. There are two cases how $c_1$ and $c_2$ can interact, either they are disjoint or they are nested. In the first case $c_1$ and $c_2$ can be reduced independently. For each cluster only a subtree of the PQ-tree has to be considered. Because these subtrees are disjoint, in the worst case the whole PQ-tree has to be traversed once per level. In the second case suppose that $c_2$ is nested in $c_1$. Then all descendants of $c_2$ are descendants of $c_1$ and the result of reducing $c_2$ can be used for reducing $c_1$. It is not necessary to traverse the corresponding PQ-subtree again but we can start the second REDUCE at the pert-root of $c_2$.

This result can be generalised to the whole cluster tree by using a simultaneous bottom up traversal of the cluster tree $\Gamma$ and the PQ-tree $T(R^i)$. After a cluster $c$ has been reduced, all PQ-leaves representing vertices contained in $c$ are consecutive in any permutation stored in $T(R^i)$. They are exactly the leaves of a pertinent subtree. Pert-root of this subtree can be a single node or a consecutive part of a Q-node. Therefore we temporarily replace the pertinent subtree(s) by a new PQ-leaf $X_c$ with label $c$. This avoids calling REDUCE for inner PQ-nodes which may not be supported by existing PQ-tree implementations. It is important that the replaced subtrees are reinserted later in the same order as they had before their removal. For this, reversions of their parent Q-node and other modifications have to be respected. Fortunately this can be done easily by remembering which sibling pointer of $X_c$ represents the direction, w. l. o. g. the first stored pointer. This is similar to the *direction indicators* of [4].

Algorithm 3 shows the method REDUCE-CLUSTERS. The cluster tree $\Gamma$ is traversed in a similar way as the REDUCE method traverses a PQ-tree. The cluster nodes are processed bottom up using a queue to ensure that nodes cannot be processed before their children on the same level have been processed. This can be tested by comparing the number of processed children with $child\_count(c, i+1)$, the number of children of $c$ on level $i+1$.

### 3.3 Correctness

**Theorem 1.** *Algorithm 1 with the extended CHECK-LEVEL method shown in Algorithm 2 returns true if and only if the graph is cl-planar.*

*Proof.* "⇒": Since our algorithm does not modify the level planarity test part, a positive result ensures that $G$ is level planar. Thus it remains to be shown that the restrictions R1–R3 imposed by the definition of cl-planarity are satisfied. The semantics of R1 for the intersection of a cluster $c \in C$ and a level line $i$ are exactly the same as the semantics of a REDUCE operation applied to the

**Algorithm 3:** REDUCE-CLUSTERS

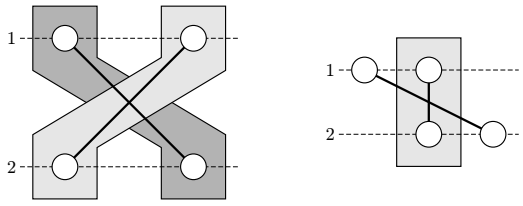
**Input**: PQ-tree $T(R^i)$, Vertices $V^{i+1}$ of the next level

**Output**: PQ-tree $T(R^i)'$ with reduced clusters

**foreach** $c \in C \cup V$ **do** $children\_leaves[c] \leftarrow \emptyset$

Initialise Queue $Q$ with $V^{i+1}$
**while** $Q$ *not empty* **do**
    $c' \leftarrow delete\_first(Q)$
    $c \leftarrow parent(c')$ *// parent in* $\Gamma$
    *// make cluster vertices consecutive*
    $T(R^i) \leftarrow$ REDUCE($T(R^i)$, $children\_leaves[c']$)
    **if** $T(R^i) = \emptyset$ **then**
        **return** $T(R^i)' \leftarrow \emptyset$
    **end**
    *// expand children*
    **foreach** $Y \in children\_leaves[c']$ **do**
        replace $Y$ by $subtrees[Y]$
    **end**
    *// contract pertinent subtree(s)*
    $X_{c'} \leftarrow$ new PQ-leaf with label $c'$
    **if** *pert-root has only pertinent children* **then**
        $subtrees[X_{c'}] \leftarrow \{pert\text{-}root\}$
    **else**
        $subtrees[X_{c'}] \leftarrow$ pertinent children of *pert-root*
    **end**
    REPLACE_PERT($T(R^i)$, $X_{c'}$)
    $insert(children\_leaves[c], X_{c'})$
    *// ensure correct processing order*
    **if** $|children\_leaves[c]| = child\_count(c, i+1)$ **then**
        $insert(Q, c)$
    **end**
**end**
**return** $T(R^i)' \leftarrow T(R^i)$

children of $c$ on level $i$. Since our algorithm explicitly calls REDUCE for every cluster on every level it is clear that R1 is satisfied. R2 is trivially satisfied for level connected graphs, because crossing clusters would imply crossing edges which are prohibited by the level planarity test. See Fig. 3(a). The same is true for R3. The graph is proper and thus the crossing edge connects two adjacent levels. Between these two levels there is an edge in the cluster because of the level connectivity of the graph. Any intersection between these two edges is prohibited by level planarity. See Fig. 3(b).




(a) Violation of R2 is not possible

(b) Violation of R3 is not possible

**Fig. 3.** Correctness of the algorithm

"$\Leftarrow$": Given a cl-planar graph we have to show that our algorithm returns true. Suppose the algorithm returns false. This means that a call of REDUCE failed, either in the level planarity test part or in REDUCE-CLUSTERS. In the former case the graph is not level planar. In the latter case there is no level planar permutation respecting R1. In any case this contradicts the assumption. □

### 3.4 Complexity

The complexity of Algorithm 3 depends also on the complexity of the *child_count* operation returning the number of cluster children on a given level. We assume that the used data structure for clustered level graphs provides $\mathcal{O}(1)$ time access to this information. If this is not true, an additional $\mathcal{O}(k|V|)$ size data structure can be pre-computed in $\mathcal{O}(k|V|)$ time. We use a two dimensional matrix $M_{ci} = \mathit{child\_count}(c, i)$ with $c \in C \cup V$ and $i \in \{1, \ldots, k\}$ as indices. $M_{ci}$ is filled as shown in Algorithm 4 which traverses the cluster tree $\Gamma$ in a similar way as Algorithm 3. Having this efficient *child_count* operation, the complexity of REDUCE-CLUSTERS derives as follows:

**Lemma 2.** *The time complexity of REDUCE-CLUSTERS described in Algorithm 3 is* $\mathcal{O}(|V|)$.

**Algorithm 4:** CHILD-COUNT

**Input**: $G = (V, E, \Gamma, \phi)$
**Output**: $M_{ci}$

Initialise $M_{ci}$ with zeros
**foreach** $v \in V$ **do** $M_{v,\phi(v)} \leftarrow 1$
**foreach** $c \in C$ **do** $processed\_children[c] \leftarrow 0$

Initialise Queue $Q$ with $V$
**while** $Q$ *not empty* **do**
  $c' \leftarrow delete\_first(Q)$
  $c \leftarrow parent(c')$ *// parent in* $\Gamma$
  **for** $i \leftarrow 1$ ***to*** $k$ **do**
    **if** $M_{c'i} > 0$ **then** $M_{ci} \leftarrow M_{ci} + 1$
  **end**
  $processed\_children[c] \leftarrow processed\_children[c] + 1$
  **if** $processed\_children[c] = |children(c)|$ **then**
    $insert(Q, c)$
  **end**
**end**

**foreach** $v \in V$ **do** $M_{v,\phi(v)} \leftarrow 0$

**return** $M_{ci}$

*Proof.* In REDUCE-CLUSTERS every cluster is considered exactly once. Since $\Gamma$ is of linear size this can be done in $\mathcal{O}(|V|)$ time. Additionally every node of the PQ-tree is considered only once such that the time complexity of the REDUCE operations sum up to $\mathcal{O}(|V|)$. □

**Theorem 2.** *There is an $\mathcal{O}(k|V|)$ time algorithm for testing cl-planarity of a level connected and proper hierarchy.*

## 4 Discussion

The given algorithm solves the cl-planarity problem only for a subclass of cl-graphs. It is desirable to extend it to general cl-graphs, but a straightforward extension is difficult.

Level planarity testing has been extended to non-hierarchical graphs in [11–16]. This is realised by the utilisation of multiple PQ-trees, one for each connected component of $G^i$. If a vertex is common to more than one PQ-tree, these are merged into one. In a straightforward extension of our algorithm clusters can span multiple PQ-trees. This means that REDUCE-CLUSTERS cannot be applied directly. A possible solution would be to additionally merge the PQ-trees according to the contained clusters. It is not clear, however, how this could be done because in contrast to vertex merges there is no distinct position in the higher PQ-tree where the smaller one is to be inserted.

An application of our algorithm to non-proper graphs leads to problems as well. A priori it is not clear whether long span edges entering or leaving a cluster have to be routed within or outside of the cluster. In Fig. 4 is not clear whether the reduction of the cluster nodes on level 2 has to include the dotted edge. When processing level 3 it becomes clear that this edge has to be routed within the cluster, but this is too late. On level 2 both routing alternatives would have to be stored in the PQ-tree. This is not possible, however, without major extensions of the data structure.

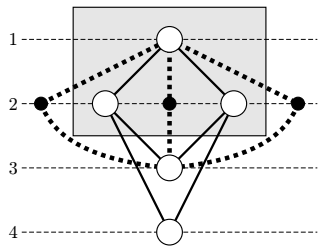



**Fig. 4.** Problems with long span edges

The third remaining restriction of our algorithm is the needed level connectivity. A straightforward idea to extend it to cl-graphs that are not level connected would be to insert level connecting dummy edges for each cluster. It is difficult, however, to find the correct places for insertion without violating cl-planarity. The same problem occurs with c-planarity. There are some advances like in [9] but the general problem is unsolved. Apparently the connectivity of the graph plays a major role for the detection of c-planarity and cl-planarity.

## 5 Conclusion

We have presented an algorithm for detecting clustered $k$-level planarity of a level connected proper hierarchy in $\mathcal{O}(k|V|)$ time. For this we have enhanced the linear time algorithm for level planarity detection of [6, 11–16].

The presented extension to level planarity testing can also be used to extend the level planar embedding algorithm of Sect. 2 to calculate a cl-planar embedding without any major modifications. Such an embedding can be used as a basis for generating a drawing using for example the algorithm of Sander [17–19]. This algorithm draws the clusters as nested rectangles, i. e., as convex regions.

It is open, whether our algorithm can be improved to linear time. This would probably be the case if the nesting relations $\Gamma_i$ of each level $i$ can be computed in linear time, such that each $\Gamma_i$ only contains the vertices of level $i$ and clusters with at least two children in $\Gamma_i$. Further investigations are desired for level graphs which are not necessarily proper and level connected hierarchies. It is not clear if this problems can be solved in polynomial time.

## References


[1] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13:335–379, 1976.

[2] R. Brockenauer and S. Cornelsen. *Drawing Clusters and Hierarchies,* In M. Kaufmann and D. Wagner, editors*, Drawing Graphs*, volume 2025 of *LNCS*, chapter 8, pages 193–227. Springer, 2001.

[3] M. Chandramouli and A. A. Diwan. Upward numbering testing for triconnected graphs. In *Proc. Graph Drawing, GD 1995*, volume 1027 of *LNCS*, pages 140–151. Springer, 1996.

[4] N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. A linear algorithm for embedding planar graphs using PQ-trees. *Journal of Computer and System Sciences*, 30:54–76, 1985.

[5] E. Dahlhaus. A linear time algorithm to recognize clustered planar graphs and its parallelization. In C. L. Lucchesi, editor, *3rd Latin American Symposium on Theoretical Informatics, LATIN '98*, volume 1380 of *LNCS*, pages 239–248. Springer, 1998.

[6] G. Di Battista and E. Nardelli. Hierarchies and planarity theory. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(6):1035–1046, 1988.

[7] Q.-W. Feng, R. F. Cohen, and P. Eades. Planarity for clustered graphs (extended abstract). In *European Symposium on Algorithms, ESA '95*, volume 979 of *LNCS*, pages 213–226. Springer, 1995.

[8] M. Forster. Applying crossing reduction strategies to layered compound graphs. In *Proc. Graph Drawing, GD 2002*, volume 2528 of *LNCS*, pages 276–284. Springer, 2002.

[9] C. Gutwenger, M. Jünger, S. Leipert, P. Mutzel, M. Percan, and R. Weiskircher. Advances in $c$-planarity testing of clustered graphs. In M. Goodrich and S. Kobourov, editors, *Proc. Graph Drawing, GD 2002*, volume 2528 of *LNCS*, pages 220–235. Springer, 2002.

[10] P. Healy and A. Kuusik. The vertex-exchange graph: A new concept for multi-level crossing minimisation. In *Proc. Graph Drawing, GD 1999*, volume 1731 of *LNCS*, pages 205–216. Springer, 1999.

[11] L. S. Heath and S. V. Pemmaraju. Recognizing leveled-planar dags in linear time. In *Proc. Graph Drawing, GD 1995*, volume 1027 of *LNCS*, pages 300–311. Springer, 1996.

[12] L. S. Heath and S. V. Pemmaraju. Stack and queue layouts of directed acyclic graphs: Part II. *SIAM Journal on Computing*, 28(5):1588–1626, 1999.

[13] M. Jünger and S. Leipert. Level planar embedding in linear time. In *Proc. Graph Drawing, GD 1999*, volume 1731 of *LNCS*, pages 72–81. Springer, 1999.

[14] M. Jünger and S. Leipert. Level planar embedding in linear time. *Journal of Graph Algorithms and Applications*, 6(1):67–113, 2002.

[15] M. Jünger, S. Leipert, and P. Mutzel. Level planarity testing in linear time. In *Proc. Graph Drawing, GD 1998*, volume 1547 of *LNCS*, pages 224–237. Springer, 1998.

[16] S. Leipert. *Level Planarity Testing and Embedding in Linear Time.* Dissertation, Mathematisch-Naturwissenschaftliche Fakultät der Universität zu Köln, 1998.

[17] G. Sander. Layout of compound directed graphs. Technical Report A/03/96, Universität Saarbrücken, 1996.

[18] G. Sander. *Visualisierungstechniken für den Compilerbau*. PhD thesis, Universität Saarbrücken, 1996.

[19] G. Sander. Graph layout for applications in compiler construction. *Theoretical Computer Science*, 217:175–214, 1999.