Graph Clustering I: Cycles of Cliques

(Extended Abstract)*

F. J. Brandenburg

Lehrstuhl für Informatik
Universität Passau
94030 Passau, Germany
email: brandenb@informatik.uni-passau.de

Abstract. A graph is a cycle of cliques, if its set of vertices can be partitioned into clusters, such that each cluster is a clique and the cliques form a cycle. Then there is a partition of the set of edges into inner edges of the cliques and interconnection edges between the clusters. Cycles of cliques are a special instance of two-level clustered graphs. Such graphs are drawn by a two phase method: draw the top level graph and then browse into the clusters. In general, it is NP-hard whether or not a graph is a two-level clustered graph of a particular type, e.g. a clique or a planar graph or a triangle of cliques. However, it is efficiently solvable whether or not a graph is a path of cliques or is a large cycle of cliques.

Introduction

Graph clustering is a new direction in graph drawing. Several winners of last year's graph drawing competition have used this technique [9]. The Circular Library in the Graph Layout Toolkit of Tom Sawyer Software [5] clusters graphs and then displays them e.g. as a circle of circles.

There is a general need for clustering techniques. As the amount of information to be visualized becomes larger, more structure is needed on top of the classical graph model. There is the need for abstraction and reduction. Flat graphs no longer suffice. When graphs become huge, classical graph drawing algorithms behave poorly or even fail. The further difficulty comes from the inherent complexity of large graphs. The computational complexity of the graph drawing algorithms is directly effected by the size of the graphs. There is a need for efficient algorithms in graph drawing. Here, graph clustering brings us a step forward. Now, time consuming graph drawing algorithms can be applied to small portions only, and the overall running time still remains satisfactory. However, this can work only, if the partition into clusters can be computed efficiently.

* Partially supported by the Deutsche Forschungsgemeinschaft, Forschungsschwerpunkt "Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen".

Several recursive clusterings have been proposed recently, such as higraphs [12], compound graphs [23], hierarchical and clustered graphs [7] and [8] or Hgraphs [14]. For directed acyclic graphs special hierarchical decompositions have been introduced in [11,20], or in [22]. These techniques impose a tree structure on top of the graphs. Ultimately, such approaches lead to graph grammars, where the derivation trees define a hierarchical structure and give raise to clusterings based on a finite set of rules, see [3,18].

Graph partitioning is a classical topic in graph theory (see e.g. [2]) and an important issue in the design of networks, VLSI, and parallel algorithms [18]. There are vertex and edge partitions of graphs, e.g. partitions into triangles, cliques or isomorphic graphs, and there are partitioning algorithms aiming at small sets of separators see, e.g., [1,18]. Partition is a classical NP-hard domain. Many of the graph partition problems stated above are well-known in the theory of NP-completeness, see GT10-GT16 in [10]. Generalized versions of NP-complete graph partition problems habe been studied in [2,6,13,15].

Our approach towards two-level clustered graphs considers graph partitions in more detail and aims at structural properties of the clusters and of the cluster graph. This graph is generally ignored. A graph is a two-level clustered graphs, or more precisely, an X-graph of Y-graphs, if the clusters from the partition of the vertices induce a Y-graph, and if the graph of the clusters is an X-graph. Examples are cycles of cliques or path of cliques. Every bipartite graph with a nonempty set of edges is an edge of two discrete graphs, and every rectangular grid is a (horizontal) path of (vertical) paths. Two-level clustered graphs are defined by graph projections. A graph projection is a mapping between graphs, which is different from other common mappings on graphs, such as graph embeddings and graph homomorphisms. In terms of generalized embeddings, graph projections have a high load, dilation one, and a shrinking expansion. As graph homomorphisms, projections are onto for the vertices and they require a membership property for the clusters.

Two-level clustered graphs have been defined in a syntactic way by the $\odot$-operation of Kratochvíl et. al. [17]. For two disjoint graphs G and G' let $G \odot G'$ be a graph obtained by the union of the sets of vertices and edges and adding an arbitrary set of matching edges between the two sets of vertices. Thus, the $\odot$-operation is nondeterministic and does not define $G \odot G'$ uniquely. In their "Stringing Lemma", Kratochvíl et. al. [17] have applied their $\odot$-operation to complete graphs, such that the graph of clusters is planar. In our terminology these graphs are the planar graphs of cliques. Planar graphs of cliques seem to be inherently nondeterministic. Their synthesis uses the nondeterministic $\odot$-operation and their recognition is NP-complete [16].

Obviously, the recognition and representation problems are the major problems for two-level clustered graphs. Given a graph G and two classes of X-graphs and Y-graphs. Then the recognition problem is, whether G is an X-graph of Y-graphs. Thus, one must find an X-graph G' and a projection of G onto G' such

that every cluster is a Y-graph. However, these problems are NP-hard, in general. This is somewhat expected, since many classical graph partition problems can be defined in these terms, such as partition into triangles, partition into cliques, etc., see [10]. Here the top-level X-graphs are arbitrary graphs, and the Y-graphs are triangles or cliques etc. Kratochvíl's result [16] restrict the top-level to the planar graphs, but still there is NP-completeness. We add some further NP-completeness results, such as the recognition of a triangle of cliques or a clique of cliques. However, there are $O(n^3)$ resp. $O(n^2)$ algorithms to decide whether or not a graph is a path of cliques or is a cycle of cliques of length at least six. The latter looks surprising, since it is NP-hard whether a graph is a triangle of cliques, i.e. a cycle of length three.

This is our first paper on graph clustering. In forthcoming papers we'll investige all relevant classes of graphs, including paths, cycles, trees, grids, planar graphs, and cliques, and vary the depth of the clustering to more than two levels.

Basic Notions

First, we recall some basic notions on graphs and establish our notation.

A graph $G = (V, E)$ consists of a finite set of vertices V and a finite set of edges E. Edges are denoted as pairs of vertices (u, v). We consider simple, undirected graphs without self-loops and multiple edges. Such edges are erased by a clean-up procedure.

The *neighbors* of a vertex u are the vertices $neigh(u) = \{v \in V | (u, v) \in E\}$. $neigh(u)$ does not include u itself. Let $neigh[u] = \{u\} \cup neigh(u)$. Accordingly, for a set of vertices U, let $neigh(U) = \{v \in V - U | (u, v) \in E$ for some $u \in U\}$ and $neigh[U] = U \cup neigh(U)$. The subgraph *induced* by a set of vertices U is denoted by $G_{|U}$. The *complement* of G is $\bar{G} = (V, \bar{E})$ with $\bar{E} = \{(u, v) | u, v \in V, u \neq v, (u, v) \notin E\}$.

A graph $G = (V, E)$ with $V = \{v_0, \ldots, v_{n-1}\}$ is a *path*, if $E = \{(v_i, v_{i+1}) | i = 0, \ldots, n-2\}$. G is a *cycle*, if $E = \{(v_i, v_{i+1}) | i = 0, \ldots, n-1 \bmod n\}$ and is the *clique* K_n, if $E = \{(v_i, v_j) | i \neq j\}$.

The size of a graph is $size(G) = n$, where $n = |V|$. For convenience, we shall assume the neighbors of a vertex and the set of neighbors neigh(U) can be computed in time proportional to the sizes of U and neigh(U). Usually, the computation time also takes the number of edges into account. Then a scan of G takes $O(n^2)$ instead of $O(n)$.

We now come to the central definition of our approach. A graph projection is a mapping between two graphs. However, it is different from other well-known mappings between graphs and imposes a two-level structure on the graphs, defined by the host graph and by the induced subgraphs.

Definition
A *projection* of a graph $G = (V, E)$ onto a graph $G' = (V', E')$ is a mapping $f : G \to G'$ such that $f : V \to V'$ is many-to-one and onto and there is an edge $(u', v') \in E'$ if and only if there are vertices $u, v \in V$ with $f(u) = u'$, $f(v) = v'$ and $(u, v) \in E$.

Since f is onto, every vertex $v' \in V'$ induces a subgraph of G defined by the set of vertices $f^{-1}(v')$. Every vertex v' resp. the set $f^{-1}(v')$ is called a *supervertex* of G and the induced subgraph $G_{|v'} = (f^{-1}(v'),\ E \cap (f^{-1}(v') \times f^{-1}(v')))$ is called a *cluster*. The supervertices define a partition the set of vertices V into pairwise disjoint sets. The set of edges E is partitioned into inner and interconnection edges. An edge $e = (u, v) \in E$ is an *inner* edge, if $f(u) = f(v)$. Then $f(e)$ becomes a self-loop, and is omitted by our convention. An edge e is an *interconnection* edge, if $f(u) \neq f(v)$. Then there is an edge $(f(u), f(v)) \in E'$ connecting the supervertices. Conversely, for every edge $(u', v') \in E'$ there is an interconnection edge $e = (u, v)$ such that $f(u) = u'$ and $f(v) = v'$. Conversely, every partition of the set of vertices V defines a mapping between graphs, which is a projection, if it is onto and has a representative for every interconnection edge.

Definition
Let $f : G \to G'$ be a projection of G onto G'. If G' is a graph from a class X and for every $v' \in V'$, the clustered graph $G_{|v'}$ is from some class Y', then G is *a an X-graph of Y-graphs*. More generally, G is called a *two-level clustered graph*.

Two-level clustered graphs can be defined in a syntactic way by the nondeterministic $\odot$-operation introduced by Kratochvíl, et al. [17]. For graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $V_1 \cap V_2 = \emptyset$ let $G_1 \odot G_2 = \{V_1 \cup V_2, E_2 \cup E_2 \cup M\}$, where $M \subseteq V_1 \cup V_2$ is an arbitrary matching. $G_1 \odot G_2$ is not uniquely determined. This operation is extended to tuples of graphs by defining $G_1 \odot (G_2 \odot \ldots \odot G_n)$.

Kratochvíl et al. have considered graphs of the form $G = G_1 \odot G_2 \odot \ldots \odot G_n$, where each G_i is a complete graph, and the graph of representatives $\tilde{G}$ is planar. $\tilde{G} = (\{G_1, \ldots, G_n\} |\ \tilde{E})$ and there is an edge (G_i, G_j) in $\tilde{E}$ if and only if $i \neq j$ and there are vertices $v_i \in V_i$ and $v_j \in V_j$ such that $\{v_i, v_j\}$ is an edge in G. Kratochvíl (personal communication) has observed, that graphs of the above form are exactly the planar graphs of cliques. In [17] it has been shown that the complete bipartite graph $K_{5,5}$ is not a planar graph of cliques and that every such graph is a string graph, i.e. the intersection graph of curves. In [16] Kratochvíl has proved the NP-compleness of the recognition problem for planar graphs of cliques. This results shall we improved to triangles of cliques.

How shall we draw two-level clustered graphs? How shall we draw a an X-graph of Y-graphs? This comes naturally with the definition. On the top level, draw the graph G' by your favorite algorithm for X-graphs. The algorithm may draw G' with variable size nodes reflecting the sizes of the induced subgraphs. We do not display the graph G as a whole; instead we browse into the vertices of G' and

display the clusters by drawing the induced subgraphs as Y-graphs. This can be extended to pairs of supervertices $u', v' \in V'$ and then displaying the clusters as Y-graphs together with the interconnection edges between the vertices of the clusters. For a nice display, the clusters should be drawn such that the endpoints of interconnection edges appear on the outer boundary of the drawn Y-graphs. Graph drawing algorithms of that kind are not yet around.

For a graph $G = (V, E)$ and graph classes X and Y the recognition problem is, whether or not G is an X-graph of Y-graphs, and to find a corresponding representation. What is the complexity of this decision problem? A solution of this question requires a partition of the vertices $V = V_1 \cup \ldots \cup V_k$ with $V_i \cap V_j = \emptyset$ such that every induced subgraph $G_{|V_i}$ is a Y-graph. The sets $\{V_1, \ldots, V_k\}$ become the vertices of the graph G'. The edges of G' must have representatives in the graph G. Hence, the set of edges E is partitioned into the sets of edges of the induced subgraphs and the set of edges representing the edges of G'. This partition is many-to-one and onto and is directed by the properties of the graph classes X and Y.

The question whether or not a graph G is an X graph of Y graphs is NP-hard for very many instances of graphs G and classes X and Y. This is particularly true, when X is the class of all graphs. In that case, the host graph G' has no particular structure and the problem of the existence of a certain projection reduces to a classical decomposition problem for graphs or to the existence of a special subgraph. The interconnection edges can be ignored, and known NP-hardness results apply, see [6, 10]. In particular, the well-known graph partition problems into triangles, isomorphic subgraphs, Hamiltonian subgraphs, forests, cliques, or perfect matchings and graph colorability (see GT11-GT16 in [10]) can be described as a graph projection onto arbitrary graphs.
Similary, if Y is the set of graphs with a single vertex, then nothing changes and all NP-hard problems remain as they were.

However, these classes of graphs seem too general and are not interesting to us. Our inspiration came from the circular layouts of [5] and we first consider cycles and paths of cliques. Surprisingly, the recognition problem can be solved in efficiently.

A further consequence of our approach will be a richer classification of graphs, particularly for graph drawing. So far, graph drawing distinguishes four classes: general graphs, directed acyclic graphs, planar graphs and trees. Other classes of graphs are not of general interest, because either they seem too special or there are no appropriate graph drawing algorithms, yet. Our approach changes this situation. And it opens a wide field for further investigations.

Cycles of Cliques

For clarity we recall the definition of cycles of cliques.

Definition
A graph $G = (V, E)$ is a *cycle of cliques* (resp. *path of cliques*), if there is a cycle (a path) $G' = (V', E')$ and a projection $f : G \to G'$ of G onto G' such that for every $v' \in V'$, the subgraph induced by $f^{-1}(v')$ is a clique.
G is a k-cycle of cliques, if G' is a cycle of size (or length) k.

Hence, there is a partition of the set of vertices $V = V_0 \cup \ldots \cup V_{k-1}$ with $V_i \cap V_j = \emptyset$ for $i \neq j$ such that for every $i, 0 = 1, \ldots, k-1$, the subgraph induced by V_i is a clique. Moreover, with the proper ordering there are vertices u, v with $u \in V_i$, $v \in V_{i+1} \pmod k$ and $(u, v) \in E$, and conversely, for every edge $(u, v) \in E, u \in V_i$ implies $v \in V_i$ or $v \in V_{i\pm1} \pmod k$.

For the decision problem it must be checked whether or not the set of edges can be partitioned into clique edges and cycle edges, where the clique edges are one-to-one and the cycle edges are many-to-one.

Cycles of cliques have a particular structure. The deletion of a supervertex cuts the cycle and leaves a path of cliques. Now, every supervertex except at the ends is a clique separator. However, not every clique separator can be used as a supervertex. There may be some nondeterminism, which cannot be decided locally. The circular saw with two defects in Fig. 1 is an example of a cycle of cliques. However, the triangles cannot be chosen as clusters. Because of the quadrilateral on top the left resp. right sides must be taken.

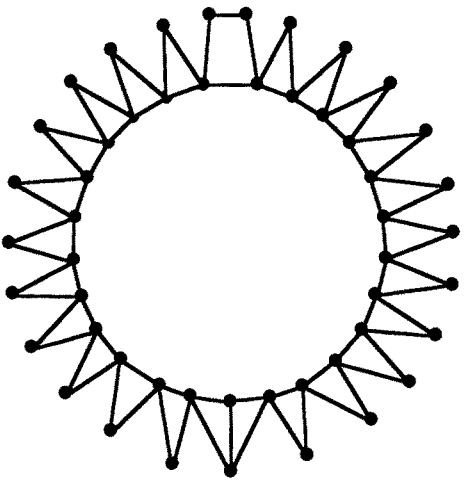

Figure 1

A partition of a graph $G = (V, E)$ into cliques is directly related to a coloring of the complement graph $\bar{G}$. A graph coloring partitions the vertices such that there are edges only between vertices from different clusters. A partition of a graph into a k-cycle of cliques implies a k-coloring of the complement graph,

such that every pair of non-adjacent clusters is a complete bipartite graph and two adjacent clusters are not complete bipartite.

It should be noted that the Circular Layouts from Tom Sawyer's Graph Layout Tool [5] are not fully captured by our definition of cycles of cliques. Their examples allow almost cycles of almost cliques. However, there is no reasonable way to define "almost" and to handle it efficiently.

A vertex and an edge of cliques can be recognized easily. The case $k = 3$ is different. Then a k-cycle is a triangle. Triangles of cliques are dense graphs, and there may be an edge between any pair of vertices. This explains why they are hard to recognize. Our NP result improves the result of Kratochvíl [16] on the NP-completeness of planar graphs of cliques.

Theorem 1
It is NP-complete, whether or not a graph G is a triangle of cliques.

Proof.
G is a triangle of cliques if and only if the complement graph $\bar{G} = (V, \bar{E})$ is 3-colorable and the induced partition $V = V_1 \cup V_2 \cup V_3$ is connected in G.
The latter holds e.g. by the reduction from 3-SAT given in [19]. Hence, a 3-SAT expression is satisfiable if and only if G is a triangle of cliques.

Using k-colorability, this result can be generalized to cliques of cliques. Notice that a clique is a clique of cliques, but not conversely.

Theorem 2
For $k \geq 3$ it is NP-complete, whether or not a graph G is a k-clique of cliques.

These results suggest, that the recognition problems for two-level clustered graphs are NP-hard, except for some trivial cases. *This is false!* There are polynomial time algorithms for paths of cliques and also for cycles of cliques, when triangles are excluded.

Theorem 3
There is an $O(n^3)$ algorithm to decide whether or not a graph G is a path of cliques and to compute the corresponding representation.

Proof.
(sketch). Suppose that $G = (V, E)$ is a path of cliques with the supervertices $V_0, \ldots, V_{k-1}$ and $V_i \subseteq V$ for $i = 0, \ldots, k-1$. Each supervertex is a clique, and there are further edges between vertices of G if and only if they are in adjacent supervertices. Thus, we order the vertices of G from V_0 to V_{k-1} and orient the edges accordingly.

By induction, suppose that a path of supervertices $V_0, \ldots, V_{i-1}$ has been computed, and a subset C_i of the next supervertex V_i is known. The vertices from $V_0, \ldots, V_{i-1}$ have been marked as "old". Then the next supervertex V_i contains

the vertices from C_i and is contained in neigh[C_i]. The remaining vertices in neigh[C_i] - V_i are a subset of the next supervertex. So we define an orientation of the edges from a marked to a yet unmarked vertex, i.e. from V_{i-1} to V_i. The vertices from V_i are marked as "old" and we proceed to the next stage and compute the next supervertex V_{i+1}.

For a set of vertices $U \subseteq V$ define $next(U) = \{w \in V | w \in neigh(U)$ and w is not marked$\}$. $next(U)$ is the set of new neighbors of U. A set of vertices $U \subseteq V$ *forms a clique*, if the induced subgraph $G_{|U}$ is a clique. Two disjoint cliques U_1 and U_2 *form a clique*, if $U_1 \cup U_2$ does. Otherwise, for every $u_1 \in U_1$ there is some $u_2 \in U_2$ such that the edge (u_1, u_2) is missing in G.

Consider a nonempty set of vertices C_i which forms a clique. C_i is taken as a subset of the current supervertex V_i. The first goal is to expand C_i into V_i. However, as we shall see, this is not unique and a decision cannot be made by local computations. See Figure 1. There the triangles do not form the proper supervertices. Hence, we compute several candidates $V_{i,j}$ for the supervertex V_i, and keep track of them and check which of them will eventually succeed to represent G as a cycle of cliques. A candidate fails, if it induces a set of vertices as a supervertex, which is not a clique. Candidates can be identified, if they behave similarly and induce cliques which differ not essentially. For example, if a vertex lies between V_i and V_{i+1} and forms a clique with both, then it can be added to either of them.

Let $C_i \subseteq V_i$ and let the vertices from $V_0, \ldots, V_{i-1}$ be marked. Then $C_i \subseteq V_i \cup next(C_i)$ and $\emptyset \neq next(C_i) \subseteq V_i \cup V_{i+1}$. Hence, the vertices from $next(C_i)$ must be partitioned into V_i and V_{i+1}. Consider the set $next(C_i)$. Since $next(C_i)$ is covered by at most two cliques, it partitions into four mutually disjoint sets C_{i+1}, R, B and Y. Some but not all of them may be empty. Each of them forms a clique. C_{i+1} and C_i do not form a clique. I.e. for every $u \in C_{i+1}$ there is some $v \in C_i$ such that (u, v) is not an edge in G. Hence, the vertices from C_{i+1} must belong to the next supervertex V_{i+1}. The sets R, B, and Y each form a clique with each of C_i and C_{i+1}. Y forms a clique with each of R and B, whereas R and B do not form a clique. A pair (u, v) is in $R \times B$ iff the edge (u, v) is missing in G. The computation of the sets C_{i+1}, R, B and Y is easily done by coloring the complement graph $\bar{G}$. Consider the subgraph of G induced by $C_i \cup next(C_i)$. Its complement must be 2-colorable, such that all vertices from C_i are colored by the same color. Otherwise, $C_i \cup next(C_i)$ must be partitioned into at least three supervertices. These are either disconneced or they branch and then cannot form a cycle. The complement graph restricted to $C_i \cup next(C_i)$ is almost discrete. However, every vertex from C_{i+1} is connected to some, but not all vertices from C_i, the "yellow" vertices from Y are isolated and the vertices from $R \cup B$ are bipartite subgraphs in $\bar{G}$, i.e. they are colored "red" and "blue". If there are several components, R or B are chosen arbitrarily.

Next, we decide the vertices from $R \cup B$, and move them into C_i and C_{i+1}. Mark the vertices from $C_i \cup next(C_i)$. For every pair u, v from $R \cup B$ with $u \in R$ and

$v \in B$, compute $next(u)$ and $next(v)$. If $next(u) \subset next(v)$, then add u to C_i and $v \cup next(u)$ to C_{i+1}. Since u and v must be in distinct supervertices, there is no other choice. If $next(u) = next(v)$, then arbitrarily add u to C_i and v and $next(u)$ to C_{i+1}. u and v can be interchanged. Finally, if $next(u)$ and $next(v)$ are pairwise incomparable, there is an error. The chosen C_i is false, and shall be ignored. If there is no alternative, then the graph G is not a path of cliques.

Now, consider the vertices from Y. Y forms a clique with each of C_i and C_{i+1}. Thus, for the vertices of Y there is a choice between the supervertices V_i and V_{i+1}. This nondeterminism cannot be solved by local computations. However, the vertices from Y can be ordered.

Mark the vertices from $C_i \cup next(C_i)$. For vertices y and y' from Y, if $next(y) = next(y')$, then identify y and y'. If $next(y)$ and $next(y')$ are pairwise incomparable, then y and y' must belong to the same supervertex, V_i or V_{i+1}. Then merge them into y" with $next(y") = next(y) \cup next(y')$. Finally, if $next(y) \subset next(y')$, then $y' \in C_i$ implies $y \in C_i$, otherwise, the next supervertex is not a clique. Hence, we can order the (subsets of) elements of Y according to the strict inclusion of their sets of next neighbors. Let $Y = (y_1, \ldots y_r)$ with $next(y_j) \subset next(y_{j+1})$. Since Y forms a clique with C_i, r is smaller than the cliquesize of G.

For the supervertex V_i we have $r + 1$ candidates, namely $V_{i,0} = C_i$ and $V_{i,j} = C_i \cup \{y_1, \ldots, y_s | s \leq j, j = 1, \ldots, r\}$. Each of them is taken into consideration and is tested in parallel in the next phases. However, if $r \leq 2$, then every $V_{i,j}$ uniquely determines its next supervertex $V_{i+1,j}$, or the number of candidates reduces, since a candidate fails or is merged with another candidate. Every candidate can be associated with a vertex of G or a supervertex. Hence there are at most $O(n)$ candidates, and each of them takes at most $O(n)$ computation steps, when we assume that neighbors can main be computed in linear time in the numbers of vertices. Thus, the decision procedure takes $O(n^2)$ time, which is $O(|V| \cdot |E|)$, if edges are taken into account.
It remains to compute a candidate for the first supervertex. Suppose that G is a k-path of cliques. The cases $k = 1$ and $k = 2$ are obvious. For $k \geq 3$ consider the set of endpoints of paths defining the diameter of G. Let C_0 be one of these sets, i.e. $C_0 = \{u|$ there is some vertex v such that the shortest path between u and v has length $diam(G)\}$, where $diam(G)$ is the diameter of G. The diameter of a graph and the set C_0 can be computed by using an "all-pairs shortest path" algorithm, and this takes $O(n^3)$.

Finally, we consider cycles of cliques for sufficiently long cycles.

Lemma
If G is a k-cycle of cliques, then the diameter of G is bounded by $diam(G) \leq 2 \cdot \lfloor k/2 \rfloor + 1$.

Notice that the graphs used for the NP-completeness result of Theorem 2 have diameter two.

Theorem 4
If $diam(G) \geq 4$, then there is an $O(n^2)$ algorithm to decide whether or not a graph G is a cycle of cliques and to compute the representation of G as a cycle of cliques.

Proof.
Suppose that G is a k-cycle of cliques with the supervertices $V_0, V_1 \ldots, V_{k-1}$ in that cyclic order. Then $k \geq 4$. The idea of the proof is to cut the cycle, say at a supervertex V_0. Then we obtain a path of cycles, and can use Theorem 5. Thus, we must compute a subset C_0 of the first supervertex V_0, and we must mark the vertices from V_{k-1} in order to move through G in circular order. These marks are crucial, because then the set of next neighbors of a supervertex is 2-colorable in the complement graph. Otherwise, and in particular for triangles, three or more colors are needed. Since $\mathrm{diam}(G) \geq 4$, there is a shortest path of length four $v_0 \to v_1 \to v_2 \to v_3 \to v_4$. Mark (or temporarily delete) v_0 and its neighbors $\mathrm{neigh}(v_0)$. If G is cycle of cliques, then the rest graph $G - (neigh[v_0'])$ is a path of at least three cliques. Now use the construction from Theorem 3 with the vertex (v_2) as subset of the first supervertex. When the supervertices including (v_4) are compputed, reinsert $\mathrm{neigh}[v_0]$.

When the cycle is closed it is checked, that the candidate chosen as the first supervertex is feasible.

The algorithm takes time $O(n^2)$, since the paths of length at least 4, the computations of C_0 and the main decision procedure can each be done in quadratic time with the assumptions made above. If a scan through a graph $G = (V, E)$ takes $O(|V| + |E|)$ time, then we obtain $O(n^3)$.

Conclusion

Graph projections and the induced graph clusterings are a field for new investigations. The partitions into cycles of cliques are a very first step. Other classes of interest are trees and planar graphs at the top level. For trees, there may be a similarity to the tree-decomposition of graphs; for planar graphs we take a look at planar graphs of planar graphs as a new attempt towards "almost planar graphs". E.g. the K_{16} can be decomposed into four K_4's and thus is a planar graph of planar graphs; however, the K_{17} does not admit such a projection.

References

1. C.J. Alpert and A.B. Kahng: Recent directions in netlist partitioning: a survey. Integration, the VLSI J. **19** (2995), 1-18.
2. J. Bosik: Decompositions of Graphs. Kluvwer Academic Publishers, Dordrecht (1990).

3. F.J. Brandenburg: Designing graph drawings by layout graph grammars. Graph Drawing 94, LNCS 894 (1995), 416-427.
4. R.C. Brewster, P. Hell and G. MacGillivray: The complexity of restricted graph homomorphisms. Discrete Mathematics **167/168** (1997), 145-154.
5. U. Dogrusöz, B. Madden and P. Madden: Circular layout in the graph layout toolkit. Graph Drawing 96, LNCS 1190 (1997), 92-100.
6. D. Dor and M Tarsi: Graph decomposition is NP-complete: proof of Holyer's conjecture. SIAM J. Comput. **26** (1997), 1166-1187.
7. P. Eades and Q.W. Feng: Multilevel visualization of clustered graphs. Graph Drawing 96, LNCS 1190 (1997), 101-112.
8. P. Eades, Q.W. Feng and X. Lin: Straight-line drawing algorithms for hierarchical graphs and clustered graphs. Graph Drawing 96, LNCS 1190 (1997), 113-128.
9. P. Eades, J. Marks and S. North: Graph-Drawing Contest Report. Graph Drawing 96, LNCS 1190 (1997), 129-138.
10. M.R. Garey and D.S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Fransisco (1979).
11. D.J. Gschwindt and T.P. Murthagh: A recursive algorithm for drawing hierarchical graphs. Technical Report CS -89-02, Williams College, Williamstown (1989).
12. D. Harel: On visual formalisms. Comm. ACM **31** (1988), 514-530.
13. P. Hell and J. Nešetřil: On the complexity of H-coloring. J. of Combinatorial Theory, Series B **48** (1990), 92-110.
14. M. Himsolt: Konzeption und Implementierung von Grapheditoren. Shaker Verlag, Aachen (1993).
15. I. Holyer: The NP-completeness of some edge partition problems. SIAM J. Comput., **10** (1981), 713-717.
16. J. Kratochvíl: String Graphs. II. Recognizing string graphs is NP-Hard. J. of Combinatorial Theory, Series B **52** (1991), 67-78.
17. J. Kratochvíl, M. Goljan and P. Kučera: String Graphs. Academia, Prague (1986).
18. T. Lengauer: Combinatorial Algorithms for Integrated Circuit Layout. Wiley-Teubner Series (1990).
19. U. Manber: Introduction to Algorithms a Creative Approach. Addison Wesley, Reading (1989).
20. E.B. Messinger, L.A. Rowe and R.R. Henry: A divide-and conquer algorithm for the automatic layout of large directed graphs. IEEE Trans. Systems Man Cybernetics **21** (1991), 1-12.
21. R. Sablowski and A. Frick: Automatic graph clustering. Graph Drawing 96, LNCS 1190 (1997), 396-400.
22. F.-S. Shieh and C.L. McCreary: Directed graphs drawing by clan-based decomposition. Graph Drawing 95, LNCS 1027 (1996), 472-482.
23. K. Sugiyama and K. Misue: Visualization of structural information: automatic drawing of compound graphs. IEEE Trans. Systems Man Cybernetics **21** (1991), 876-892.
24. R.E. Tarjan: Decomposition by clique separators. Discrete Math. **55** (1985), 221-232.
25. S.H. Whitesides: An algorithm for finding clique cut-sets. Inf. Proc. Letters **12** (1981), 31-32.