NP-Completeness of Some Tree-Clustering Problems

Falk Schreiber* and Konstantinos Skodinis**

University of Passau,
94032 Passau, Germany
{schreiber,skodinis}@informatik.uni-passau.de

Abstract A graph is a tree of paths (cycles), if its vertex set can be partitioned into clusters, such that each cluster induces a simple path (cycle), and the clusters form a tree. Our main result states that the problem whether or not a given graph is a tree of paths (cycles) is NP-complete. Moreover, if the length of the paths (cycles) is bounded by a constant, the problem is in P.

1 Introduction

Graph drawing defines one of its tasks as drawing graphs in a 'nice' and 'understandable' way. But the meaning of the notions 'nice' and 'understandable' depends on the point of view; there is no universal definition of a 'good' graph layout.

One approach to produce expressive drawings are clustering techniques [6, 7, 14]. *Clustering* is a partitioning of the vertex set of a graph G into smaller sets called clusters, which satisfy certain criteria. Drawing graphs according to such a clustering appears in many application areas as VLSI design, software engineering, and knowledge representation. The main problem remains to find the corresponding clusters. Unfortunately in general this problem is NP-hard. However, if the clusters are known, 'nice' and 'understandable' graph drawings can often be constructed efficiently.

In this paper we deal with certain graph classes called *two-level clustered graphs* introduced by Brandenburg [3]. Given such a graph one is interested in drawings which make its structure transparent, especially if the graph is huge. Intuitively, given two graph classes X (level 1) and Y (level 2), a graph G is an X-graph of Y-graphs (X *of* Y graph) if the vertices of G can be partitioned into clusters, such that (i) every cluster induces a graph of class Y in G and (ii) the clusters form a graph of class X in G. The X-graph is obtained from G by shrinking first the clusters into single vertices and then the multiple edges into one edge. Examples are paths of paths, paths of cliques, trees of paths, trees

* The work of the author was supported by the German Research Association (DFG) grant BR 83576-3

** The work of the author was supported by the German Research Association (DFG) grant BR 835/7-1

S.H. Whitesides (Ed.): GD'98, LNCS 1547, pp. 288–301, 1998.
© Springer-Verlag Berlin Heidelberg 1998

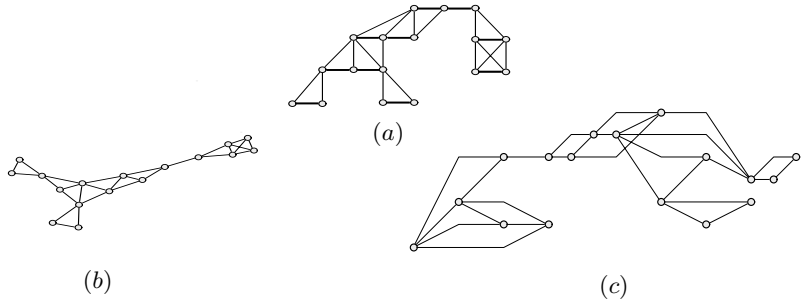

Figure1. Drawings of a *tree of paths*

of cycles and so on. It is easy to see that every $n \times m$-grid is a path of paths and every $2n \times 2n$-grid is a path of cycles. To the contrary every clique with more than four vertices is neither a path of paths nor a tree of paths.

Two-level clustered graphs have been defined in a syntactic way by Kratochvíl et. al. [12]. They can be drawn according to their structure: On the top level draw the X-graph and on the second level the Y-graphs (see [3] for more details). The obtained drawing reflects the nature of the graph and considerably simplifies its understanding. An example for a *tree of paths* is shown in Figure 1(a) with thick lines indicating the paths. The drawings (b) and (c) are generated by the algorithms from [8] and [15] respectively.

The main problem is to determine whether or not a graph is a two-level clustered graph. Complexity considerations concerning two-level clustered graphs are not known to be investigated in literature. To the contrary there is a great quantity of complexity results concerning for example partitioning of graphs, see GT11-GT16 in [9], or covering of graphs, see GT17 and GT18 in [9]. This research is still in progress [4, 5]. In fact almost all nontrivial partitioning and covering problems of graphs are NP-complete. Partitionings and coverings are two-level clusterings in which the first level graphs (X-graphs) are arbitrary. Hence, two-level clustering is a restrictive version of partitioning and covering. The restriction can lead to removing the NP-completeness. For example the problem whether or not a graph is a partition into cliques is NP-complete (see GT15 in [9]), whereas the problems whether or not a graph is a path of cliques or a large cycle of cliques are in P, as shown in [3]. Thus, it can not always be expected that every nontrivial two-level clustering problem is NP-complete.

In this paper we consider trees of paths and trees of cycles. We show that given a graph G the question whether or not G is a tree of paths (cycles) is NP-complete. Moreover, if the length of the paths (cycles) is bounded, we give a polynomial algorithm based on the dynamic programming technique to solve these questions. Additionally, it also outputs such a tree clustering, if the answer is positive. A consequence of our results is that the problem whether or not a given graph is a tree of triangles is in P. This contrasts the fact that the problem

whether or not a given graph can be partitioned into triangles is NP-complete, see GT11 in [9].

2 Basic Notions

In this section we review some basic notions on graphs and establish our notation. A *graph* with vertex set V and edge set E will be denoted by $G = (V, E)$. The cardinality of the vertex set will be called the *size* of G. We deal with undirected graphs without loops and multiple edges. The subgraph of a graph G *induced* by a vertex set U is denoted by $G[U]$. A vertex set S is a *separator* of G if graph $G[V - S]$ is disconnected.

A *path* of length n is a graph with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set $E = \{(v_i, v_{i+1}) | 1 \leq i \leq n-1\}$. Similarly, a *cycle* of length n is a graph with vertex set $V = \{v_1, \ldots, v_n\}$ with $n \geq 3$, and edge set $E = \{(v_i, v_{i+1}) | 1 \leq i \leq n-1\} \cup \{(v_n, v_1)\}$. For convenience, we also say that a subset $P \subseteq V$ is a path (cycle) in G, if the subgraph $G[P]$ is a path (cycle). Further a separator S is a *path (cycle) separator* if S is a path (cycle).

Next we define two-level clustered graphs. In order to do this we first define for a graph $G = (V, E)$ its *clustering* with respect to a partition of V.

Definition 1. *Let $G = (V, E)$ be a graph and $V_1, \ldots, V_m$ a (nonempty) partition of the vertex set V. The graph with vertex set $\{w_1, \ldots, w_m\}$ and edge set $\{(w_i, w_j) \mid u \in V_i, u' \in V_j, i \neq j,$ and $(u, u') \in E\}$ is called the clustering of G with respect to the partition $V_1, \ldots, V_m$.*

Intuitively, the *clustering of* G is constructed from G by first shrinking every subgraph $G[V_i]$ into a single vertex w_i and then merging multiple edges.

Definition 2. *Let $G = (V, E)$ be a graph and X and Y two graph classes. Graph G is an X-graph of Y-graphs (X of Y graph), if there is a partition $V_1, \ldots, V_m$ of the vertex set V, such that*

1. *the subgraphs $G[V_1], \ldots, G[V_m]$ are from the class Y, and*
2. *the clustering of G with respect to the partition $V_1, \ldots, V_m$ is from the class X.*

In this paper we restrict ourselves to *tree of paths* and *tree of cycles*. Every such graph G has a *tree clustering* T. For reasons of readability we distinguish between *nodes* of T and *vertices* of G. Every node of T represents a path (cycle) in G. We will identify a node and its representing path (cycle) in G. A branch of T at w is the maximal subtree of T containing w and exactly one neighbour of w in T. Notice that the number of branches of T at w equals the degree of w in T. If the paths (cycles) of a tree clustering T have maximal length k then T will be denoted by T^k.

As examples notice the following facts: Obviously, every tree is a tree of paths, and every $2n \times 2n$-grid is a tree (path) of cycles. To the contrary every clique with more than 6 vertices is neither a tree of paths nor a tree of cycles. Furthermore, every cycle of length n is a tree (path) of paths, but it has no tree clustering T^k of paths with $k < \lceil n/2 \rceil$.

3 Main Results

In this section we deal with the problems whether or not a graph G is a tree of paths (ToP problem) and whether or not G is a tree of cycles (ToC problem). The problems are formally defined as follows:

Name: ToP problem.		*Name*: ToC problem.
Instance: A graph G.	and	*Instance*: A graph G.
Question: Is G a tree of paths?		*Question*: Is G a tree of cycles?

If the length of the paths, respectively the length of the cycles is bounded by some integer k, then we call these problems *k-ToP*, respectively *k-ToC*.

For our purpose we need a variant of the ONE-IN-THREE 3SAT problem (see [9], L04) in which the instances meet the following requirements: (*i*) no clause contains opposite literals, and (*ii*) if a variable x appears more than once, any of its literals x and $\overline{x}$ appears in at least two clauses. We call this restricted variant the r-ONE-IN-THREE 3SAT problem.

Proposition 1. *The r-ONE-IN-THREE 3SAT problem is NP-complete.*

Proof. The proof is a simple reduction from the "positive" ONE-IN-THREE 3SAT problem, which is also NP-complete, see [9]. In this problem the instances are restricted to contain only clauses with positive literals. Let ϕ be such an instance. For every variable x appearing more than once in ϕ we add the clauses $(x \vee a_{x_1} \vee a_{x_2})$, $(x \vee a_{x_3} \vee a_{x_4})$, $(\overline{x} \vee a_{x_5} \vee a_{x_6})$, and $(\overline{x} \vee a_{x_7} \vee a_{x_8})$ to ϕ, where $a_{x_1}, \ldots, a_{x_8}$ are new variables. By construction the obtained expression ϕ' meets the desired requirements. Moreover it is easy to show that ϕ is satisfiable if and only if ϕ' is satisfiable (in the sense of ONE-IN-THREE 3SAT).

3.1 Tree of Paths

In this subsection we first show that the ToP problem is NP-complete. Then we give a polynomial algorithm solving the k-ToP problem. The algorithm can be modified easily to output a tree clustering T^k if it exists.

The key to prove the NP-completeness is the gadget $\widehat{C}$ shown in Figure 2. $\widehat{C}$ has an important property which is summarized by the next lemma.

Lemma 1. *Let G be a tree of paths, T a tree clustering of G, and $\widehat{C}$ the graph shown in Figure 2. If $\widehat{C}$ is an induced subgraph of G then the vertices of $\widehat{C}$ belong to exactly two different paths P_1 and P_2 of T, such that P_1 contains two vertices of x, y, z and P_2 contains the vertices l, r and the remaining vertex of x, y, z.*

Proof. An easy inspection shows that the vertices x, y, and z may neither belong to the same path nor to three different paths of T. Thus x, y, and z belong to exactly two different paths P_1 and P_2 of T. Without loss of generality we assume that P_1 contains x and y and path P_2 contains z, see Figure 2(a).

Now consider vertex l. Let P_3 be the path of T containing l. Obviously, P_3 is different to P_1. If P_3 is also different to P_2, then the paths P_1, P_2, and P_3 would

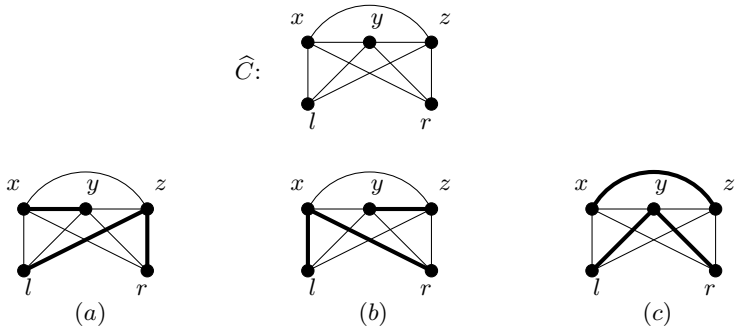

Figure2. Graph $\widehat{C}$ and its different path clusterings

not form a tree because they are pairwise connected by edges from G. Hence, P_3 and P_2 are identical in T and l belongs to P_2. For symmetry the same holds for vertex r.

Theorem 1. *The ToP problem is NP-complete.*

Proof. Obviously the ToP problem is in NP. We show that it is NP-hard. The proof is a reduction from the r-ONE-IN-THREE 3SAT problem. Let ϕ be an instance with n clauses $C_1, \ldots, C_n$. We construct a graph $G(\phi)$ and we show that ϕ is satisfiable if and only if $G(\phi)$ is a tree of paths.

The key to our construction is the graph $\widehat{C}$ of Figure 2. We create n copies $\widehat{C}_1, \ldots, \widehat{C}_n$ of $\widehat{C}$ side by side, one for every clause C_i. For convenience we will denote the copies of vertices x, y, z, l, r in $\widehat{C}_i$ by x_i, y_i, z_i, l_i, r_i, respectively. Vertices x_i, y_i and z_i correspond to the literals of the clause C_i. We create a new vertex b and connect it with all vertices l_i and r_i. Then we add an edge between vertices r_i and l_{i+1}, $1 \leq i \leq n-1$. Finally, we add an edge between two vertices in different copies if and only if the vertices correspond to opposite literals. Figure 3 illustrates the construction.

We claim that ϕ is satisfiable if and only if $G(\phi)$ is a tree of paths. In proof, suppose that there is a truth assignment satisfying ϕ. Thus, every clause has two false and one true literal. In every copy $\widehat{C}_i$ the vertices corresponding to the false literals are a path in $G(\phi)$, denoted by P_i. The vertices l_i and r_i and the vertices corresponding to the true literals in the clauses are also a path in $G(\phi)$, denoted by P. Finally, the vertex b is a trivially single path denoted by P'. By construction P' is connected with P which itself is connected with every P_i. Since there is no connection between two different paths P_i and P_j (two false literals can never be opposite and thus never be adjacent), $P_1, \ldots, P_n, P$, and P' form a tree clustering of $G(\phi)$.

Conversely, suppose that $G(\phi)$ is a tree of paths whith a tree clustering T. Our aim is to define a truth assignment for ϕ. First, take a closer look at the structure of $G(\phi)$ to explore some important information about its clustering T.

$$\phi = (x \vee y \vee z) \wedge (\overline{x} \vee u \vee v) \wedge (x \vee w \vee x) \wedge (\overline{x} \vee q \vee h)$$

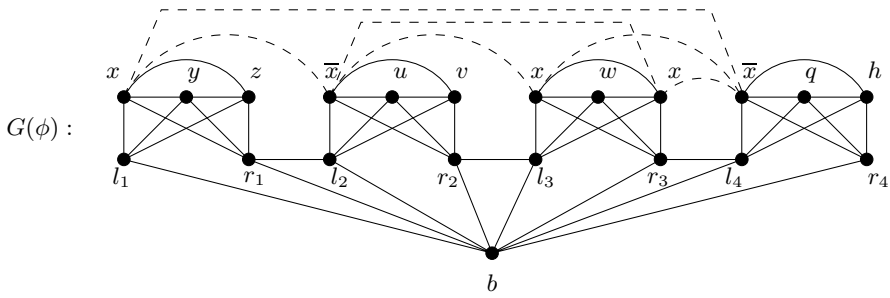

Figure3. Reduction to ToP. The dotted edges connect vertices corresponding to opposite literals.

By Lemma 1 the vertices x_i, y_i, z_i, l_i, r_i of $\widehat{C}_i$ belong to two different paths of T, say P_1^i and P_2^i, such that P_1^i contains two vertices from x_i, y_i, z_i and P_2^i contains the remaining vertices. We show that all paths P_2^i are identical in T. Suppose that P_2^i and P_2^j are different in T. Then the vertex b must belong to some of them, otherwise P_2^i, P_2^j and the path containing b are pairwise connected and they can not form a tree. But this is a contradiction, because b is connected with at least two vertices of P_2^i and with at least two vertices of P_2^j. Hence, T has one path consisting of the vertices l_i, r_i and of exactly one vertex from x_i, y_i, z_i, $1 \leq i \leq n$. We denote this path by P. Consequently, the vertex b forms its own trivial path in $G(\phi)$, denoted by P'. Additionally, P and P' are different from every path P_1^i for $1 \leq i \leq n$, of T.

Next we show by contradiction, that there are no connections between two paths P_1^i and P_1^j in $G(\phi)$. Suppose that there were a connection between some vertex $p_i \in P_1^i$ and some vertex $p_j \in P_1^j$. By construction p_i and p_j correspond to opposite literals, say x and $\overline{x}$, respectively. We directly obtain: (i) the paths P_1^i and P_1^j are the same path in T; otherwise P_1^i, P_1^j and P were pairwise different and pairwise connected in T which is a contradiction, and (ii) the variable x appears more than once.
By (ii) there are two further vertices $p_s \in \widehat{C}_s$ and $p_t \in \widehat{C}_t$, such that p_s corresponds to x and p_t to $\overline{x}$. Consider p_s and p_t, which are adjacent by construction. Their adjacency implies that they may not be both in P; otherwise P would not be a path. Thus, one of p_s and p_t, say p_t, belongs to some path, say P_1^t, different from P. Since p_i and p_t are connected in $G(\phi)$ (they correspond to opposite literals) P_1^i and P_1^t are different in T; otherwise p_i would have three neighbours on the common path, namely p_j, p_t and one in P_1^i. But then P_1^i (containing p_i and p_j), P_1^t (containing p_t) and P are pairwise different, and pairwise connected in T, which is a contradiction. Hence, there exist no connections between two

paths P_1^i and P_1^j in $G(\phi)$ and the tree clustering T of $G(\phi)$ contains exactly the paths P_1^i, $1 \leq i \leq n$, P and P'.

Now we define a truth assignment A of ϕ as follows. Let $\hat{x}$ be a literal of variable x.
We set:

$$A(\hat{x}) = \begin{cases} T & \text{if } P \text{ contains a vertex corresponding to } \hat{x} \\ F & \text{if some } P_1^i \text{ contains a vertex corresponding to } \hat{x}. \end{cases}$$

It remains to show that A is well defined. Suppose there exists a literal $\hat{x}$ having both values T and F. There are three cases:
In the first case, P contains two vertices corresponding to opposite literals. But then these vertices are adjacent in $G(\phi)$ and P would not be a path, which is a contradiction.
In the second case, $P_1^1, \ldots, P_1^n$ contain vertices corresponding to opposite literals. But this is impossible, because no P_1^i contains opposite literals, since no clause C_i contains opposite literals and because there are no connections between $P_1^1, \ldots, P_1^n$.
In the third case, P contains a vertex p and some P_1^i contains p' both corresponding to the same literal, say x. We show that this is also impossible: Consider the vertex p'' of $G(\phi)$ corresponding to literal $\overline{x}$; such a vertex exists, because the variable x appears more than once. If p'' is in P then p and p'' are adjacent in P, which is a contradiction. If p'' is not in P then it is in some path P_1^j, which contradicts to the fact that there are no connections between P_1^i and P_1^j and no clause contains opposite literals. Hence, A is well defined and the proof is complete.

Next we show that the k-ToP problem is solvable in polynomial time. Recall that the k-ToP problem is the question whether or not a given graph has a tree clustering T^k whose paths have a length bounded by some integer k.

Remark 1. Although every graph having a tree clustering T^k is a partial $2k$-tree, there is no closer relationship between the class of graphs having a tree clustering T^k and the class of partial k-trees. In fact, every cycle with length n is a partial 2-tree (independent of n), but it has no tree clustering T^k with $k < \lceil n/2 \rceil$.

In the following, we give a polynomial algorithm solving this problem. It uses a dynamic programming technique and it is similar to the algorithm given in [1] for the recognition of partial k-trees. Its idea is based on the following lemmas:

Lemma 2. *Let k be an integer and $G = (V, E)$ a graph with more than $2k$ vertices. Then G has a tree clustering T^k if and only if there is a path separator P of G, such that every connected component of $G[V - P]$ augmented by P has a tree clustering rooted by P.*

Proof. Suppose that G has a tree clustering T^k. Since G has more than $2k$ vertices, T^k has at least three paths. One of them, say P, is a separator of G. Consider the branches of T^k at P. Obviously the branches are tree clusterings of the connected components of $G[V - P]$ augmented by P.

For the only-if-case, let P be a path separator of G, and let $C_1, \ldots, C_l$ be the connected components of $G[V - P]$. Suppose that every augmented component $G[C_i \cup P]$, $1 \leq i \leq l$, has a tree clustering T_i^k rooted by P. By identifying the vertices of P in $T_1^k, \ldots, T_l^k$ we obtain a tree clustering T^k for the whole graph G.

The next lemma shows how to determine whether an augmented component $G[C_i \cup P]$ has a tree clustering rooted at a path P. Here only tree clusterings of augmented components with a smaller size than $G[C_i \cup P]$ are needed.

Lemma 3. *Let k be an integer, $G = (V, E)$ a graph, and P a path separator of G with length at most k. Let C be a connected component of $G[V - P]$ and $G[C \cup P]$ be the augmented component of C. Then $G[C \cup P]$ has a tree clustering rooted at P if and only if*

1. *C is a path in G of length at most k, or*
2. *there exist both a path separator P' of G with length at most k and connected components $C'_1, \ldots, C'_m$ of $G[V - P']$ partitioning $C - P'$, such that*
   (a) *the augmented components $G[C'_1 \cup P'], \ldots, G[C'_m \cup P']$ have tree clusterings rooted at P', and*
   (b) *P' contains all neighbours of P in C.*

Proof. Let P be a path separator of G with length at most k, C a connected component of $G[V - P]$, and $G[C \cup P]$ the augmented component of C. Assume $G[C \cup P]$ has a tree clustering T^k rooted at P. If T^k has two nodes then C necessarily is a path in G. If T^k has more than two nodes, consider the son P' of P in T^k (notice that P has only one son in T^k, otherwise C would be disconnected). P' is a path separator of G of length at most k. Obviously all connected components of $G[V - P']$ contained in C partition $C - P'$. Let $C'_1, \ldots, C'_m$ be these connected components of $G[V - P']$ contained in C. Furthermore the augmented components $G[C'_1 \cup P'], \ldots, G[C'_m \cup P']$ have tree clusterings rooted at P'. To see this, observe that every augmented component $G[C'_i \cup P']$, $1 \leq i \leq m$, corresponds to exactly one branch of T^k at P' and that every branch represents a tree clustering rooted at P'. Finally, P' contains all neighbours of P in C, since P' is the only son of P in T^k.

Conversely, first suppose that a connected component C of $G[V - P]$ is a path in G. The tree clustering consisting of root P and leaf C is a tree clustering of $G[C \cup P]$. Next suppose there exist a path separator P' of G with length at most k, and connected components $C'_1, \ldots, C'_m$ of $G[V - P']$ partitioning $C - P'$ and meeting the requirements of (a) and (b). Let $T_1^k, \ldots, T_m^k$ be the tree clusterings of the augmented components $G[C'_1 \cup P'], \ldots, G[C'_m \cup P']$ all rooted at P'. Consider the tree clustering T^k obtained by identifying the nodes representing P' in every tree clustering T_i^k, and by adding a new root P connected to P'. Since all neighbours of P in C are contained in P', T^k is a tree clustering of $G[C \cup P]$ rooted at P.

Now we present our algorithm. We can assume that the input graph has more than $2k$ vertices. First we construct the set $\mathcal{S}$ of all path separators of G with

length at most k. Then for every path separator P from $\mathcal{S}$ we find all connected components C of $G[V-P]$ and we add tuple $(G[C \cup P], P)$ to a set $\mathcal{A}$. Next we sort the elements of $\mathcal{A}$ in increasing order according to the size of their first component. Finally, we examine all graphs $G[C \cup P]$ of all tuples $(G[C \cup P], P)$ from $\mathcal{A}$, from smallest to largest and using Lemma 3 we determine whether or not graphs $G[C \cup P]$ have a tree clustering rooted at P. To save this decision we define an array I realizing the boolean function $\mathcal{A} \rightarrow \{true, false\}$. Finally, using Lemma 2 we determine whether or not graph G has a tree clustering T^k.

ALGORITHM

Input: A graph $G = (V, E)$.
Output: $answer =$ "yes" *or* "no".

begin
{ Initialization }
$\mathcal{S} := \emptyset$; $\mathcal{A} := \emptyset$; $answer :=$ "no";

{ compute set $\mathcal{S}$ }
for every $P \subseteq V$ with $|P| \leq k$
do if (P is a path separator of G)
then $\mathcal{S} := \mathcal{S} \cup \{P\}$;

{ compute set $\mathcal{A}$ and initialize array I }
for every $P \in \mathcal{S}$
do for every connected component C of $G[V-P]$
do begin
$\mathcal{A} := \mathcal{A} \cup \{(G[C \cup P], P)\}$;
$I\Big[(G[C \cup P], P)\Big] := false$;
end;
sort elements of $\mathcal{A}$ in increasing order according to the size of their first component;

{ examine every element $(G[C \cup P], P)$ of $\mathcal{A}$ in increasing order and determine whether graph $G[C \cup P]$ has a tree clustering rooted at P using Lemma 3 }
for every $(G[C \cup P], P)$ of $\mathcal{A}$ in increasing order
do begin
if (C is a path of length at most k in G)
then $I\Big[(G[C \cup P], P)\Big] := true$;
else for every P' from $\mathcal{S}$
do if ($\exists\ C'_1, \ldots, C'_m$ of $G[V-P']$ partitioning $C - P'$)
then if ($I\Big[(G[C'_i \cup P'], P')\Big] = true$ for every $1 \leq i \leq m$) and

(P' contains all neighbours of P in C)
then $I\Big[(G[C \cup P], P)\Big] := true$;
end;

{ determine whether graph G has a tree clustering using Lemma 2 }
for every P from $\mathcal{S}$
do if ($I\Big[(G[C \cup P], P)\Big] = true$ for every C of $G[V - P]$)
then begin
$answer :=$ "yes";
exit;
end;
end.

Theorem 2. *Let k be a fixed integer. The algorithm correctly determines whether or not an input graph with size n has a tree clustering T^k in $O(n^{2k+3})$ time.*

Proof. The correctness of the algorithm follows directly from Lemmas 2 and 3. We examine its complexity. The construction of set $\mathcal{S}$ takes $O(n^{k+2})$ time. There are at most $O(n^k)$ subsets of V with at most k elements and it takes at most $O(n^2)$ time to check if such a subset is a path separator of G. Similarly, the construction of $\mathcal{A}$ and initialization of I take at most $O(n^{k+2})$ time. To determine whether an augmented component $G[C \cup P]$ of G has a tree clustering rooted by P, takes at most $O(n^{k+2})$ time. There are at most $O(n^k)$ separators P' and at most n connected components of $G[V - P']$. In addition, it takes at most $O(n^2)$ time to check whether some of them partition $C - P'$. The remaining if-conditions of the loop can be checked in $O(n)$ time. Thus, to examine all $O(n^{k+1})$ tuples of $\mathcal{A}$ can be done in $O(n^{2k+3})$ time. Finally, to determine whether or not graph G has a tree clustering T^k takes at most $O(n^{k+1})$ time. Hence, the total time complexity of the algorithm is $O(n^{2k+3})$.

Our algorithm does not construct a tree clustering of the input graph, if there exists one. But this can be achieved by an obvious modification.

3.2 Tree of Cycles

In this subsection we investigate the ToC problem. First we show that it is NP-complete by a reduction from the r-ONE-IN-THREE 3SAT problem. The idea is very similar to the one used in Theorem 1. Here the key to our reduction is graph $\widetilde{C}$ shown in Figure 4. It is the graph $\widehat{C}$ in Figure 2 of the previous subsection, extended by a new vertex t connected with vertices x, y, and z. Similar to Lemma 1 we obtain:

Lemma 4. *Let G be a tree of cycles, T a tree clustering of G, and $\widetilde{C}$ the graph shown in Figure 4. If $\widetilde{C}$ is an induced subgraph of G and the vertex t has degree*

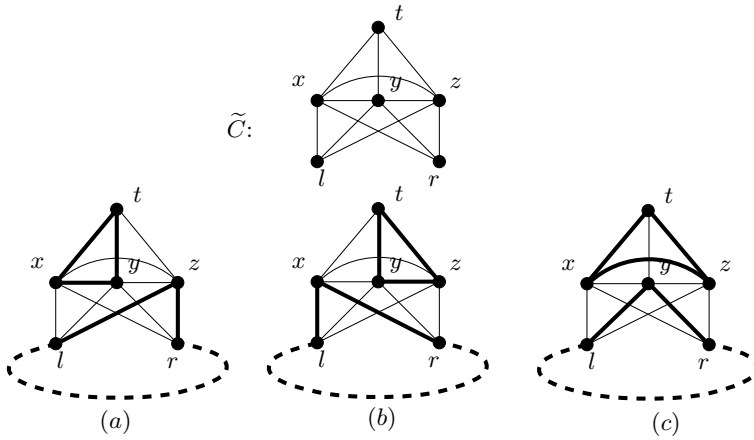

Figure4. Graph $\widetilde{C}$ and its different cycle clusterings

3 *in* G*, then the vertices* t, x, y, z, l, r *of* G *belong to two different cycles* Q_1 *and* Q_2 *of* T*, such that* Q_1 *contains vertices* t *and two of the vertices* x, y, z*, and* Q_2 *contains the remaining vertices.*

Proof. Consider the vertices x, y, and z. They may neither belong to the same cycle nor to three different cycles of T: In the first case vertex t alone would form a cycle (notice t has degree 3 in G), which contradicts to the definition of cycles. In the second case T would be no tree clustering, since x, y, and z are pairwise connected. Thus x, y, and z belong to exactly two cycles Q_1 and Q_2. Without loss of generality we assume that Q_1 contains x and y and Q_2 contains z, see Figure 4(a).

Consider the vertex t in G. Since its degree is 3, it has to belong to the cycle Q_1 (t alone can not form a cycle in T). Now consider the vertex l. Let Q_3 be the cycle of T containing l. Trivially, Q_3 is different to Q_1. If Q_3 is also different to Q_2 in T then Q_1, Q_2, and Q_3 are pairwise connected in G and T would be no tree. Hence, Q_3 and Q_2 are identical in G and l belongs to Q_2. A similar argument shows that vertex r belongs also to cycle Q_2.

Theorem 3. *The ToC problem is NP-complete.*

Proof. Obviously, the ToC problem is in NP. In order to show its NP-hardness we reduce the r-ONE-IN-THREE 3SAT problem to it. Since the reduction is very similar to the one in Theorem 1, we only sketch our proof.

Let ϕ be an instance of r-ONE-IN-THREE 3SAT. We construct a graph $G(\phi)$ as follows: We create n copies $\widetilde{C}_1, \ldots, \widetilde{C}_n$ of $\widetilde{C}$ one for every clause C_i. We identify vertices x_i, y_i and z_i of the i-th copy with the literals of clause C_i. Then we connect vertices r_i and l_{i+1}, $1 \leq i \leq n-1$, as well as vertices l_1 and r_n.

$$\phi = (x \vee y \vee z) \wedge (\overline{x} \vee u \vee v) \wedge (x \vee w \vee x) \wedge (\overline{x} \vee q \vee h)$$

$G(\phi)$:

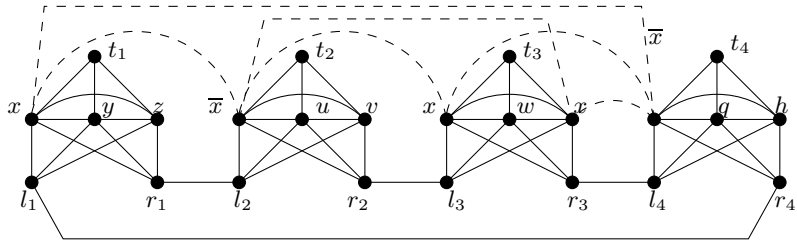

Figure5. Reduction to ToC. The dotted edges connect vertices corresponding to opposite literals.

Finally, if two vertices correspond to opposite literals then we connect these by an edge. Our construction is shown in Figure 5.

It remains to show that ϕ is satisfiable if and only if $G(\phi)$ has a tree clustering T. Suppose that there is a truth assignment satisfying ϕ. Let Q_i be the cycle of the copy $\widetilde{C}_i$, $1 \leq i \leq n$, containing vertex t_i and the vertices corresponding to false literals. Furthermore let Q be the cycle of $G(\phi)$ containing all vertices l_i, r_i, $1 \leq i \leq n$, and the vertices corresponding to the true literals in the clauses. Since there is no connection between two different cycles Q_i and Q_j (two false literals can never be opposite and thus never be adjacent), Q and Q_i form a tree clustering T of $G(\phi)$.

Conversely, suppose that $G(\phi)$ has a tree clustering T. By Lemma 4 the vertices t_i, x_i, y_i, z_i, l_i and r_i of $\widetilde{C}_i$ belong to two different cycles Q_1^i and Q_2^i of T, such that Q_1^i contains both vertex t_i and two vertices of x_i, y_i, z_i, and Q_2^i contains both the vertices l_i, r_i and the remaining vertex of x_i, y_i, z_i. We show that all cycles Q_2^i are identical in T. It is sufficient to show that for every i Q_2^i and Q_2^{i+1} are the same in T. Consider the vertex r_i of Q_2^i. It must have two neighbours in Q_2^i. These neighbours have to be l_{i+1} and one of x_i, y_i, z_i, because of Lemma 4 the remaining two neighbours of r_i have to belong to Q_1^i which is different to Q_2^i. Hence, all cycles Q_2^i, $1 \leq i \leq n$, represent the same cycle in T, denoted by Q. Consequently, T consists of the cycles Q and Q_1^i, $1 \leq i \leq n$.

Now we define a truth assignment A of ϕ as follows. Let $\hat{x}$ be a literal of variable x.
We set:

$$A(\hat{x}) = \begin{cases} T & \text{if } Q \text{ contains a vertex corresponding to } \hat{x} \\ F & \text{if some } Q_1^i \text{ contains a vertex corresponding to } \hat{x}. \end{cases}$$

Since there are no connections between two cycles Q_1^i and Q_1^j in $G(\phi)$ we can show analogously to the proof of Theorem 1 that assignment A is well defined.

Replacing paths by cycles in the algorithm for the k-ToP problem (see subsection 3.1) leads to an algorithm solving the k-ToC problem.

4 Conclusion

In this paper we have shown that given a graph G, the problem whether or not G is a tree of paths (cycles) is NP-complete. Moreover, if the length of the paths (cycles) is bounded by some integer k, we have developed an $O(n^{2k+3})$ algorithm to solve this problem. Of course our algorithm is not practical. In further investigations we will try to improve this result by finding an algorithm having a better time complexity. Besides we are interested in the complexity of the problem whether or not a given graph is a path of paths, in particular if the number of the paths is two.

Acknowledgment

We would like to thank Prof. Dr. Franz-Josef Brandenburg for his improvements of the representation.

References

[1] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:277–384, 1987.

[2] F. T. Boesch and J. F. Gimpel. Covering the points of a digraph with point-disjoint paths and its application to code optimization. *J. Assoc. Comput. Mach.*, 24(2):192–198, 1977.

[3] F. J. Brandenburg. Graph clustering I: Cycles of cliques. In G. DiBattista, editor, *Proc. of Graph Drawing*, volume 1353 of *Lect. Notes in Comput. Sci.*, pages 158–168. Springer-Verlag, New York/Berlin, 1997.

[4] E. Cohen and M. Tarsi. NP-completeness of graph decomposition problems. *J. Complexity*, 7:200–212, 1991.

[5] D. Dor and M. Tarsi. Graph decomposition is NP-complete: A complete proof of holyer's conjecture. *SIAM J. Comput.*, 26(4):1166–1187, 1997.

[6] P. Eades and Q.-W. Feng. Multilevel visualization of clustered graphs. In S. North, editor, *Proc. of Graph Drawing*, volume 1190 of *Lect. Notes in Comput. Sci.*, pages 101–112. Springer-Verlag, New York/Berlin, 1996.

[7] Q.-W. Feng, R. F. Cohen, and P. Eades. How to draw a planar clustered graph. In D.-Z. Du, editor, *Computing and Combinatorics*, volume 959 of *Lect. Notes in Comput. Sci.*, pages 21–30. Springer-Verlag, New York/Berlin, 1995.

[8] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability; A guide to the theory of NP-completeness.* W. H. Freeman, 1979.

[10] M. Himsolt. The graphlet system. In S. North, editor, *Proc. of Graph Drawing*, volume 1190 of *Lect. Notes in Comput. Sci.*, pages 233–240. Springer-Verlag, New York/Berlin, 1996.

[11] I. Holyer. The NP-completeness of some edge-partition problems. *SIAM J. Comput.*, 10(4):713–717, 1981.
[12] J. Kratochvíl, M. Goljan, and P. Kučaera. String graphs. Technical report, Academia Prague, 1986.
[13] C. Papadimitriou. *Computational Complexity.* Addison-Wesley Publishing Company, 1994.
[14] T. Roxborough and A. Sen. Graph clustering using multiway ratio cut. In G. DiBattista, editor, *Proc. of Graph Drawing*, volume 1353 of *Lect. Notes in Comput. Sci.*, pages 291–296. Springer-Verlag, New York/Berlin, 1997.
[15] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical systems. *IEEE Transactions on Systems, Man and Cybernetics*, 11:109–125, 1981.