Train Tracks and Confluent Drawings

Peter Hui[1], Marcus Schaefer[1], and Daniel Štefankovič[2]

[1] Department of Computer Science, DePaul University, Chicago, Illinois 60604
phui@students.depaul.edu, mschaefer@cti.depaul.edu
[2] Department of Computer Science, University of Chicago, Chicago, Illinois 60637
stefanko@cs.uchicago.edu

Abstract. Confluent graphs capture the connection properties of train tracks, offering a very natural generalization of planar graphs, and – as the example of railroad maps shows – are an important tool in graph visualization. In this paper we continue the study of confluent graphs, introducing strongly confluent graphs and tree-confluent graphs. We show that strongly confluent graphs can be recognized in **NP** (the complexity of recognizing confluent graphs remains open). We also give a natural elimination ordering characterization of tree-confluent graphs which shows that they form a subclass of the chordal bipartite graphs, and can be recognized in polynomial time.

1 Introduction

The area of graph drawing deals with the visualization of graphs, where the visualization meets certain aesthetic or technical constraints [1]. Typically, the goal of the drawing of a graph is to minimize some parameter such as the crossing number, or, for grid drawings, the area, the number of times an edge bends, or the total length of the edges. Among these parameters, the crossing number has probably drawn the most attention. A crossing number of zero corresponds to planarity, for which linear time algorithms are known, but, in general, determining the crossing number of a graph is **NP**-complete [4], making it a hard parameter to minimize. Recently, Dickerson, Eppstein, Goodrich, and Meng [2] suggested an extension of the notion of planarity called confluency that, while allowing crossings, hides them in the drawing. At the core is an idea similar to the train tracks of Thurston [6]: we allow edges in the drawing to merge, like train tracks, into a single track. The merging device is called a switch. Figure 1 shows how to draw complete graphs and complete bipartite graphs confluently.

Dickerson, Eppstein, Goodrich, and Meng [2] identified several classes and families of graphs which are confluent, including interval graphs and cographs. They also gave examples for graphs which are not confluent (their smallest example is obtained from the Petersen graph by removing a single vertex), and a heuristic algorithm to recognize whether a graph is confluent or not. Interestingly, they did not study the complexity of the recognition problem.

The main contribution of this paper is to show that a natural strengthening of confluency in graphs can be recognized in **NP**. In Section 2 we define the

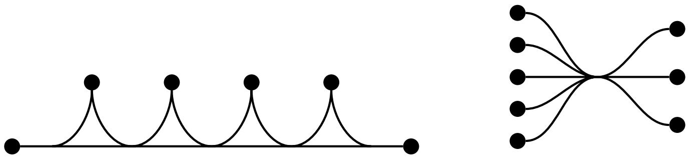

Fig. 1. How to draw K_6 and $K_{5,3}$ confluently.

notions of confluency and strong confluency. Their relationship is investigated in Section 3 by studying their underlying train tracks. Section 4 shows that strong confluency can be recognized in **NP** by giving a polynomial upper bound on the number of switches necessary to represent a graph. We think it is not unlikely that the problem will turn out to be **NP**-complete.

If confluency does turn out to be **NP**-hard, it will be of interest to identify large, and natural, subclasses which can be recognized efficiently. One immediate way of obtaining interesting classes of confluent graphs is by taking well-known graph classes whose definition depends on planarity, and replace planarity with confluency. In that manner we obtain outer-confluent graphs (see Section 6), and tree-confluent graphs, whose confluent drawings are treelike. In Section 5 we will show that the tree-confluent graphs can be recognized efficiently with the help of an elimination order characterization.

2 Train Tracks and Confluent Drawings

Definition 1. *A* curve *is a continuous mapping of* $[0,1]$ *into the Euclidean plane; we often identify a curve and its image. A curve is* smooth*, if it is differentiable (intuitively, it cannot make sharp turns). A smooth curve which does not self-intersect is called* locally monotone *[2].*

Definition 2. *A* train track drawing with vertices V and switches S *is a subset* D *of the Euclidean plane such that* (i) $V \cap S = \emptyset$, (ii) *there is a injective mapping of* $V \dot\cup S$ *into* D *(we identify a point in* $V \dot\cup S$ *with its image),* (iii) *any curve in* D *not containing a switch must be smooth, and* (iv) *any two overlapping curves in* D *must have a common tangent at any point of overlap; that is, they have to join smoothly.*

A curve in a train track drawing which shares exactly its two endpoints with $V \dot\cup S$ *is called a* branch.

Based on this notion of a train track drawing, we derive two graph drawing concepts.

Definition 3. *We call a graph* $G = (V, E)$ confluent*, if there is a train-track drawing* D *on* V *such that* $uv \in E$ *if and only if there is a locally monotone curve in* D *with endpoints* u *and* v *that does not contain any other points of* V*. In this case we call* D *a* confluent drawing *of* G*.*

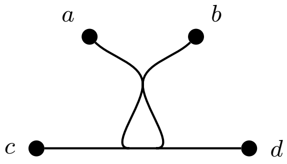

Fig. 2. K_4 or $K_4 - e$?

For an example, consider the train track drawing in Figure 2. We can easily trace locally monotone curves connecting all pairs of vertices – with the exception of a and b. There is a smooth curve connecting a to b, but it is not locally monotone, since it has to self-intersect. Hence, the train track drawing in Figure 2 is a confluent drawing of a $K_4 - e$.

When tracing a train track drawing visually, the requirement to avoid self-intersections seems to force a reader to backtrack to determine whether two points are connected. Removing this requirement leads to the following notion.

Definition 4. *We call a graph $G = (V, E)$* strongly confluent*, if there is a train track drawing D on V such that $uv \in E$ if and only if there is a smooth curve in D with endpoints u and v that does not contain any other points of V. In this case we call D a* strongly confluent drawing *of G.*

Using this new definition, we would say that the train track drawing in Figure 2 is a strongly confluent drawing of a K_4.

Remark 1. The notion of confluency was introduced by Dickerson, Eppstein, Goodrich, and Meng in [2]; at a first glance it might seem that confluency is a stronger requirement than strong confluency. The opposite, however, is true; every strongly confluent graph is confluent (as we will see in Corollary 1), and there is a confluent graph which is not strongly confluent.

By definition, any point of D at which several curves combine is a switch. A switch has two sides, each with an arbitrary number of incoming curves. Every such switch can be replaced by a series of simple switches, where a *simple switch* is a switch in which two curves merge into a single curve. For example, the drawing of K_6 in Figure 1 uses simple switches, whereas the drawing of $K_{5,3}$ in the same figure uses a single switch which is not simple. Figure 3 shows how to draw $K_{5,3}$ using only simple switches.

For the rest of the paper we will use switch synonymously with simple switch, unless explicitly stated otherwise.

3 Train Tracks

We want to capture the combinatorial structure of a train track drawing D in graph-theoretic terms, abstracting from the particular embedding. To this end, we call a vertex-labelled graph $H = (V \dot\cup S, F, o)$ a *train track* if the following holds:

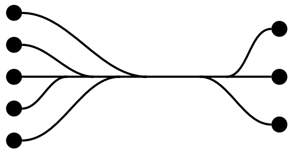

Fig. 3. How to draw $K_{5,3}$ using only simple switches.

(*i*) The vertex set of H consists of two types of vertices V and S, we call *vertices* and *switches* of H,
(*ii*) switches have degree 3,
(*iii*) $o(s) \in V \dot\cup S$ is one of the three neighbors of s (for every switch $s \in S$).

Remark 2. We will often consider H as a directed graph, for example to specify in which direction the undirected edge uv is traversed. In that case we will write (u, v) or (v, u), and we will tacitly consider the graph as symmetric; that is, for every directed edge, the reverse edge also belongs to the graph.

We think of $o(s)$ as determining the orientation of the switch s: if we enter the switch s coming from $o(s)$, it forks into two branches. A curve in a train track drawing now corresponds to a walk in the train track which *respects the orientation of the switches* in the sense that for every part (u, s, v) of the walk s is a switch, and $o(s)$ is either u or v (and u and v are different from each other). We call such a walk *acceptable*. We can now rephrase our notions of confluency and strong confluency in terms of train tracks.

Lemma 1. *A graph $G = (V, E)$ is confluent if there is a planar train track H such that $uv \in E$ if and only if there is an acceptable path from u to v in H. The graph is strongly confluent if there is a planar train track H such that $uv \in E$ if and only if there is an acceptable walk from u to v in H.*

Proof. Consider a train track drawing D with vertices V and switches S. Construct a train track H as follows: $V \dot\cup S$ will make up the vertex set of H. Include an edge (u, v) in F if in D there is a curve from u to v which does not cross through any vertices or switches. We assumed that switches are simple, hence there are three branches leaving s. Let $o(s)$ be the endpoint (other than s) of the edge corresponding to the branch that extends the other two branches smoothly. Then H is a planar train track in which every acceptable walk corresponds to a smooth curve in D, and every acceptable path to a locally monotone curve. ■

Remark 3. Given a train track, the graph it represents can be found in polynomial time. In the case of strong confluency this is obvious, for confluency the problem can be reduced to a matching problem [3].

Theorem 1. *If $G = (V, E)$ is strongly confluent, then it is represented by a planar train track $H = (V \dot\cup S, F, o)$ such that $ab \in E$ if and only if there is an acceptable path P_{ab} in H from a to b.*

We omit the proof of Theorem 1. Together with Lemma 1 it immediately implies a relationship between the two notions of confluency we introduced.

Corollary 1. *Any strongly confluent graph is confluent.*

The inclusion is strict, consider for example the graph drawn in Figure 4. By adding some more vertices and edges on the outside, we can force that all the switches occur within the circle. The resulting graph is confluent, but not strongly confluent (since vertex 6 will always be connected to vertex 4 in a strongly confluent drawing).

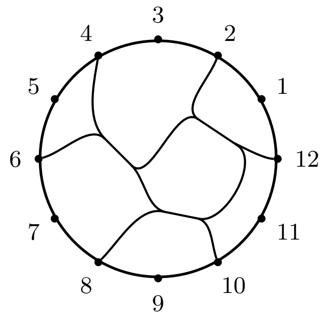

Fig. 4. Construction for graph which is confluent but not strongly confluent.

4 Strong Confluency in NP

Lemma 2. *If $G = (V, E)$ is confluent, then there is a train track H representing G, and acceptable paths P_e for every edge $e \in E$ such that the following condition holds:*

> *If P is a longest path contained in both some P_e and some P_f, then P is a single edge.*

Proof. We need a measure of overlap between two paths P_e and P_f. To this end, we introduce the numbers

$$o_{ef} := \sum_{P \text{ maximal subpath of } P_e \cap P_f} |P|^2.$$

With this we can establish the following claim:

> Suppose H is chosen to minimize the number o_{ef}. In that case, if P is a path contained in both P_e and P_f, then P is a single edge.

If the conclusion of the claim is false, there is a path (u, x, v) belonging to both P_e and P_f. Let y be the endpoint of the third edge incident to x; without loss of generality, we can assume that $o(x) = u$. Since P_e and P_f are paths, the edge xy cannot belong to either of them. Modify H as follows: remove edges ux and xv and add two new vertices u', v' and edges uu', $u'v'$, $v'v$, $u'x$ and xv'; set $o(u') = u$, $o(v') = v$, and $o(x) = u'$. Modify P_e and P_f such that one of them uses

(u, u', x', v', v) and the other (u, u', v', v). This will split the maximal common subpath of $P_e \cap P_f$ containing (u, x, v) into two parts. Since $(i + j)^2 > i^2 + j^2$ for $i, j \geq 1$, this strictly reduces o_{ef} showing that H did not minimize it. This establishes the claim.

In the modification made to H in the proof of the claim, we can route any other P_g through (u, u', v', v) if it used (u, x, y) or through (u, u', x, y) if it used (u, x, v); in either case the length of another P_g path will be increased by at most one.

More importantly, if any maximal subpath of P_g and P_h is an edge, then the modification to H will not change that: if P_g and P_h were affected by the modification and shared a single edge on the vertices u, x, v, and y, it must have been ux, and one of P_g and P_h must have used xv and the other xy; hence, after the modification they will only share uu'.

Let $e_1f_1, e_2f_2, \ldots, e_kf_k$ be an ordering of all pairs of distinct edges of G. The above observation immediately implies that if we choose H so as to minimize (in lexicographic ordering) the vector

$$(o_{e_1f_1}, o_{e_2f_2}, ..., o_{e_kf_k})$$

then any paths P_e and P_f intersect in isolated edges. ■

For the rest of this section we will concentrate on strongly confluent graphs. Because of Corollary 1, we can still apply Lemma 2 in that case, concluding that overlap between a P_e and a P_f consists of non-adjacent edges. Moreover, these overlaps between P_e and P_f correspond to crossings in a planar drawing of H. That is, if we have the path (u_e, s, t, v_e), part of P_e and (u_f, s, t, v_f), part of P_f then u_e and v_e cannot be on the same side of st in the planar drawing of H, since otherwise we could have reduced o_{ef} by having two separate paths (u_e, v_e) and (u_f, v_f) as shown in Figure 5.

There is one scenario which would prohibit the application of the move shown in Figure 5, namely if there was a third P_g making use of the edge st. This, however, is not possible, since one pair from P_e, P_f, P_g would share a path of length ≥ 2.(Note that this operation would not be valid if the representation was just confluent, since lifting the path could introduce new connections between vertices not possible before.)

Our goal is to show that we can assume the number of switches in H to be polynomial in the number of vertices. To this end we equip the train track with an edge labelling that contains connectivity information.

Given a train track $H = (V \dot\cup S, F, o)$ for $G = (V, E)$ we define a labelling of the directed edges of H as follows:

$$\ell(u, v) = \{a \in V : \text{ there is an acceptable walk from } a \text{ to } v \text{ in } H \\ \text{passing through } (u, v)\}.$$

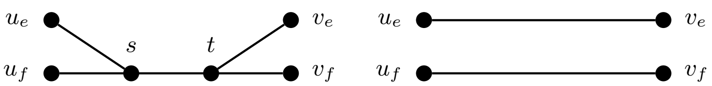

Fig. 5. Lifting a path (in a strongly confluent representation).

From the definition it follows that ℓ is the minimal labelling fulfilling

(i) $a \in \ell(a, u)$ for any edge $(a, u) \in F \cap (V, V \dot\cup S)$,

(ii) $N_G(a) = \bigcup_{(u,a)\in F} \ell(u, a)$ for any $a \in V$, where $N_G(a) = \{b : ab \in E\}$ is the neighborhood of a in G,

(iii) for any switch $s \in S$ and its neighbors $u = o(s)$, v, w:

$$\ell(u, s) \subseteq \ell(s, v) \cap \ell(s, w), \text{ and}$$
$$\ell(s, u) \supseteq \ell(s, v) \cup \ell(w, s).$$

By the results proved so far we know that if $G = (V, E)$ is strongly confluent, then there is $H = (V \dot\cup S, F, o)$ such that $ab \in E$ if and only if there is an acceptable path P_{ab} from a to b in H.

Lemma 3. *If $G = (V, E)$ is strongly confluent, then it is represented by a train track H with $O(|V|)^6$ vertices and switches.*

Let $uv \in E$. Consider a path P_{uv} in H whose inner vertices are all switches, and the function $\ell(\overrightarrow{e})$ as we move the directed edge $\boldsymbol{e}$ along P_{uv} from u to v. This yields a monotone function, namely, if $\boldsymbol{e}$ occurs before $\boldsymbol{f}$ on P_{uv} and both edges are directed towards v, then $\ell(\overrightarrow{e}) \subseteq \ell(\overrightarrow{f})$. Therefore, $\ell(\overrightarrow{e})$ can take on at most $|V|+1$ different values along P_{uv}. Similarly, if we move an edge $\overleftarrow{e}$ directed towards u along P_{uv} from u to v, the corresponding label sets are monotonously decreasing, and, hence, also take at most $|V| + 1$ different values along P_{uv}. Consequently, the expression $(\ell(\overrightarrow{e}), \ell(\overleftarrow{e}))$ can change value less than $2(|V|+1)$ times as we travel along P_{uv} from u to v.

For each $uv \in E$ we color those switches at which $(\ell(\overrightarrow{e}), \ell(\overleftarrow{e}))$ changes red, and the remaining switches blue. Note that at most $(2|V| + 1)|E|$ switches are colored red. We call the maximal segments of P_{uv} which do not contain red switches blue segments. There are at most $4(|V| + 1)|E|$ blue segments. We will show that there is a drawing such that any two blue segments intersect at most once. Hence there is a drawing with at most $(2|V|+1)|E|+2(4(|V|+1)|E|)^2 = O(|V|^6)$ switches.

Consider two edges e and f in H that are adjacent crossings of a blue segment P with other blue segments. There are two possible scenarios depending on whether the crossings are parallel or not (as earlier, the sharp angles represent the forking part of a switch, and thus define o). Figure 6 shows how the order of two parallel crossings can be swapped.

We can use a similar move for non-parallel crossings, as shown in Figure 7.

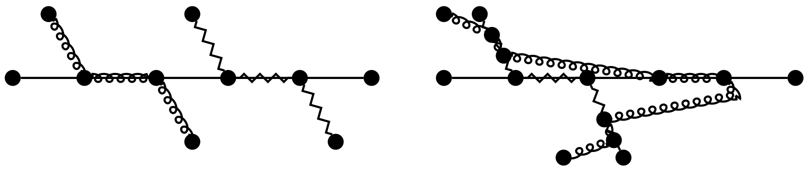

Fig. 6. How to swap two parallel crossings.

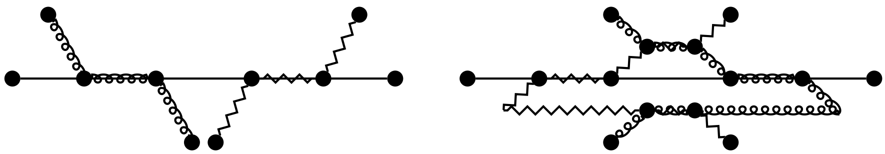

Fig. 7. How to swap two non-parallel crossings.

Note that in both cases we can extend the labelling of H to the newly introduced edges so that the graph represented by H remains the same: we simply label the new edges with $(\ell(\overrightarrow{e}), \ell(\overleftarrow{e}))$.

Suppose that two blue segments P and R cross more than once. Let e_1, e_2 be crossings of P and R such that there are no crossings of P and R between e_1 and e_2 on R. There may be crossings of P and R between e_1 and e_2 on P. Label them by i if after cutting R between e_1 and e_2 they would be in the same component of R as e_i. There is a pair of neighboring crossings f_1, f_2 labeled by $1, 2$, respectively. Using the swap moves on edges intersecting R we can make e_1, e_2 adjacent on R and then shortcut P eliminating half of the intersections created by the swap moves. Similarly using the swap moves on edges intersecting P we can make f_1, f_2 adjacent on P and then shortcut R eliminating half of the intersection created by the swap moves. In one of the cases we decrease the total number of intersections while preserving the property that any two paths intersect in paths of length 1. Hence there is a train track in which any two blue segments intersect at most once.

Corollary 2. *Strong confluency can be tested in* **NP**.

Proof. Corollary 2 shows that if G is strongly confluent, then there is a train track representing G of size polynomial in $|G|$. In **NP** we can guess any such train track, and verify that it represents G. ■

5 Tree-Confluent Graphs

We call a train track drawing D *tree-like*, if it does not contain a closed curve (the curve would not have to be smooth or locally monotone). For example, Figure 8 shows a tree-like train track drawing. On the other hand, Figure 1 shows a train track drawing representing K_6 which is not tree-like. We call a confluent graph which can be represented by a tree-like train track drawing *tree-confluent* (the strong case being the same). We will see later that all tree-confluent graphs are bipartite, so there is no tree-like train track drawing representing K_6.

In graph theoretic terms, the underlying abstract train track of a tree-confluent graph has to be a tree. A graph is tree-confluent if and only if it is represented by a planar train track which is a tree. We now give a characterization of tree confluent graphs in terms of a vertex elimination ordering. This characterization leads to a fast recognition algorithm.

Theorem 2. *A graph is tree confluent if and only if repeatedly removing (i) vertices of degree 1, and (ii) vertices u such that there is another vertex v with $N(u) = N(v) \neq \emptyset$, leads to the trivial graph (containing only a single vertex).*

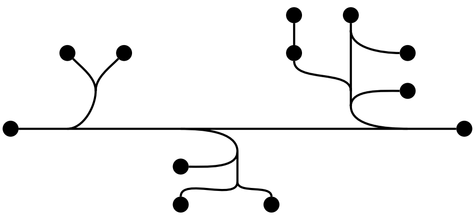

Fig. 8. A tree-like train track drawing.

Proof. First observe that if G is tree confluent then it will still be tree confluent after the removal of vertices of type (i) or (ii). Furthermore, if G is not tree confluent, it cannot become tree confluent by removing a vertex of type (i) or (ii): if $G - \{v\}$ were tree confluent and v has degree 1 in G, then it has degree 1 in the underlying train track, hence G is tree confluent; similarly if $G - \{v\}$ is tree confluent, and G contains another vertex u with $N(u) = N(v)$, then we can replace u in the train track for $G - \{v\}$ by a switch that branches to u and v, showing that G is tree confluent (note that G does not contain the edge uv, since $N(u) = N(v)$).

Since the trivial graph is tree confluent, this observation implies that any graph which can be reduced to the trivial graph by removing vertices of type (i) and (ii) is tree confluent.

Furthermore, for the other direction, the observation shows that the order of removal is irrelevant, and it is sufficient to show that if G is tree confluent, there is some order $\mathcal{E}$ in which to remove vertices of type (i) and (ii) such that we end up with the trivial graph.

Suppose $G = (V, E)$ is tree confluent; then there is a planar train track $H = (V \dot\cup S, F, o)$ which represents G. Consider F as a set of directed edges. We define a function p from F to $\mathbb{N}$ as follows:

$$p(u, v) = |\{w \in V : \text{there is a walk from } v \text{ to } w \text{ that does not use } u\}|.$$

Figure 9 shows an example of p.

Using p we define a second function r from S to $\mathbb{N}$. Let the neighbors of s be u, v, w, then

$$r(s) = \min\{p(s, u) + p(s, v), p(s, u) + p(s, w), p(s, v) + p(s, w)\}.$$

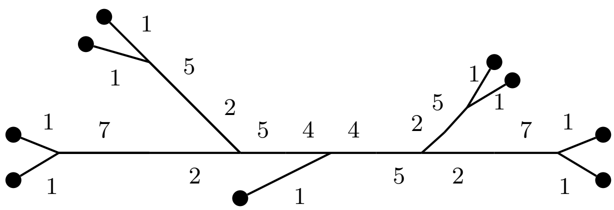

Fig. 9. A train track. Every edge is labelled with the number of vertices contained in the portion of the train track to which the edge points.

We begin the construction of $\mathcal{E}$ by repeatedly removing vertices of degree 1 in G until there are no such vertices left. If G has not turned into the trivial graph at that point, it has to contain at least one switch (otherwise it would be a tree on its vertices and would have to contain a vertex of degree 1).

Choose the switch s with minimal $r(s)$. We claim that $r(s) = 2$. First note that $r(s) \geq 2$, because p is always at least 1. Now suppose that $r(s) > 2$. Then s must be adjacent to at least one edge su such that $p(s, u) \geq 2$, since otherwise $r(s)$ would equal 2. The portion of the train track to which su leads cannot contain any switches, for if it contained a switch t, then $r(t)$ would be strictly less than $r(s)$ violating the minimality of $r(s)$. Therefore, the vertices in the portion of H to which (s, u) leads must form a tree. However, the leaves of this tree would have degree 1, and we already eliminated all those vertices.

Let s be a switch with $r(s) = 2$. Then there are u and v such that $p(s, u) + p(s, v) = 2$, and therefore $p(s, u) = p(s, v) = 1$. Hence, u and v have to be vertices of G. Since neither of them can have degree 1 in G, the switch s must be oriented to fork into u and v which implies that $N(u) = N(v)$. Hence we can continue the construction of $\mathcal{E}$ by selecting u, for example.

We continue in this fashion, eliminating vertices of degree 1 as long as possible, and then identifying leaf vertices of switches s with $r(s) = 2$. We showed that the only reason such a switch would not exist is that the graph G has turned into the trivial graph, which is what we had to show. ■

The elimination characterization of Theorem 2 leads to a randomized linear time algorithm for recognizing tree-confluent graphs.

In [5], Golumbic and Goss introduced the now well-known class of graphs known as the *chordal bipartite graphs*, which are those bipartite graphs in which every cycle of length at least 6 contains a chord (that is, no cycle of length at least 6 is induced). Removing a vertex of degree 1 or a vertex u such that there is another vertex v for which $N(u) = N(v)$ from a graph does not change the property of a graph being chordal bipartite. This observation gives us the following lemma.

Lemma 4. *Every tree confluent graph is chordal bipartite.*

The reverse is not true as witnessed by a C_6 with a single chord.

6 Open Problems

While we have shown that strong confluency can be recognized in **NP**, we currently have no such result for confluency. Although the two notions are very similar, their combinatorial nature seems to be quite different. At this point we cannot even rule out the possibility that a confluent graph needs an exponential number of switches to be realized (although that would not necessarily affect membership in **NP**, as witnessed by the example of string graphs [7]).

Identifying large classes of confluent graphs remains a challenging task. We suggest the notion of outer-confluency (confluent graphs that can be drawn in a disk with all the vertices on the boundary of the disk). As in the case of confluency there are examples of graphs that are outer-confluent but not strongly outer-confluent (see Figure 4). Dickerson, Eppstein, Goodrich, and Meng [2] showed – in effect – that all cographs are outer-confluent (even strongly outer-confluent), thereby also showing that outer-confluency is a strict superclass of tree-confluency. It does not seem unlikely that outer-confluent graphs can be recognized in polynomial time.

We seem to have a good understanding of tree-confluent graphs; the main missing piece is a deterministic linear time recognition algorithm.

References

1. Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.
2. Matthew T. Dickerson, David Eppstein, Michael T. Goodrich, and Jeremy Yu Meng. Confluent drawings: visualizing non-planar diagrams in a planar way. In *Proc. 11th Int. Symp. Graph Drawing (GD 2003)*, Lecture Notes in Computer Science. Springer-Verlag, 2003.
3. David Eppstein, 2004. personal communication.
4. Michael R. Garey and David S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983.
5. M. Golombic and C. F. Goss. Perfect elimination and chordal bipartite graphs. *J. Graph Theory*, 2:155–163, 1978.
6. R. C. Penner and J. L. Harer. *Combinatorics of train tracks*, volume 125 of *Annals of mathematics studies*. Princeton University Press, 1992.
7. Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Recognizing string graphs in NP. *J. Comput. System Sci.*, 67(2):365–380, 2003. Special issue on STOC2002 (Montreal, QC).