Train Tracks and Confluent Drawings

Peter Hui[1], Michael J. Pelsmajer[2], Marcus Schaefer[3], and Daniel Štefankovič[4]

[1] Department of Computer Science, DePaul University, Chicago, Illinois 60604
phui@students.depaul.edu
[2] Department of Applied Mathematics, Illinois Institute of Technology, Illinois 60616
pelsmajer@iit.edu
[3] Department of Computer Science, DePaul University, Chicago, Illinois 60604
mschaefer@cti.depaul.edu
[4] Computer Science Department, University of Rochester, Rochester, New York 14627-0226 stefanko@cs.rochester.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, and show that this class coincides with the $(6, 2)$-chordal bipartite graphs. Finally, we define outerconfluent graphs, and identify the bipartite permutation graphs as a natural subclass.

1 Introduction

The area of graph drawing is concerned with visualizing graphs meeting certain aesthetic or technical constraints [2]. 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 [8], making it a hard parameter to minimize. Recently, Dickerson, Eppstein, Goodrich, and Meng [5] 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 [10]: 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, by example, how to draw complete graphs and complete bipartite graphs confluently.

Dickerson, Eppstein, Goodrich, and Meng [5] 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

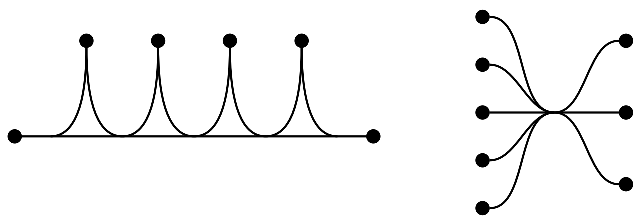

Fig. 1. How to draw K_6 and $K_{5,3}$ confluently.

heuristic algorithm to recognize whether a graph is confluent or not. Interestingly, they did not study the complexity of the recognition problem.

The first main contribution of this paper is to show that a natural strengthening of confluency can be recognized in **NP**. In Section 2 we define the 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 graph classes whose definition depends on planarity, and replace planarity with confluency. In that manner we obtain tree-confluent and outerconfluent graphs. Our second main contribution is the study of tree-confluent graphs: graphs whose confluent drawings are treelike. In Section 5 we give an elimination order characterization for tree-confluent graphs, and show that tree-confluency is equivalent to several other well-known graph properties. In Section 6 we initiate the study of outerconfluent graphs, defining the natural subclass of outerconfluent bipartite graphs, which turn out to be exactly the bipartite permutation graphs.

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 *(or* embedded*)* [5].

Definition 2. *A* train track drawing with vertices V and switches S *is a closed subset D of the Euclidean plane such that*

- *V and S are disjoint,*

– *there is an injective mapping of $V \dot{\cup} S$ into D (we identify a point in $V \dot{\cup} S$ with its image),*
– *any curve in D not containing a switch must be smooth,*
– *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.

The main difference between a train track drawing and a traditional drawing lies in the interpretation of adjacency and the presence of switches. The notion of a train track drawing allows us to 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.*

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 $K_4 - e$.

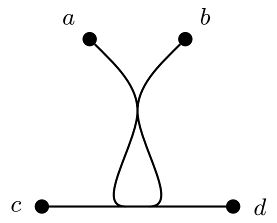

Fig. 2. K_4 or $K_4 - e$?

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 [5]; 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.

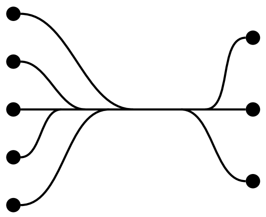

Fig. 3. How to draw $K_{5,3}$ using only simple switches.

For the rest of the paper we will use "switch" to mean simple switch unless explicitly stated otherwise.[5]

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 the triple $H = (V \mathbin{\dot\cup} S, F, o)$ a *train track* if the following holds:

(i) The nodes of H are of two types, the *vertices* V and the *switches* S,
(ii) switches have degree 3,
(iii) F is the set of *edges* of H (*branches* in the drawing),
(iv) o maps S to $V \mathbin{\dot\cup} S$ such that $o(s)$ is one of the three neighbors of s.

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[6] in the train track which *respects the*

[5] In the topological literature on train tracks, simple switches are called *trivalent* or *generic*.
[6] A *walk* is a sequence of vertices adjacent in the order of the sequence, with repetitions allowed. A *path* is a walk without repeating vertices.

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: $V \dot\cup S$ are the nodes of H. Include an edge uv in the edge-set F of H if in D there is a curve from u to v which does not pass through any vertices or switches. We assumed that switches are simple, hence there are three branches leaving each 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 2. 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 [6].

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.*

Proof. If $G = (V, E)$ is strongly confluent, then there is a planar train track $H = (V \dot\cup S, F, o)$ representing it.

We modify H slightly. Consider a switch $s \in S$, and let its neighbors be u, v, and w such that $o(s) = u$.

We call the switch s *ambiguous*, if there is an acceptable walk from s to s which begins with $(s, o(s))$ and ends with $(o(s), s)$. Construct a new train track H' from H as follows: replace every ambiguous switch s of H connected to u, v, and w by three switches as shown in Figure 4 allowing immediate passage from u, v, and w to any other among them.

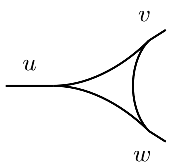

Fig. 4. An ambiguous switch replaced by three switches.

We claim that there is an acceptable walk from a to b in H if and only if there is an acceptable path from a to b in H'. This proves the statement of the theorem.

First, suppose there is an acceptable path from a to b in H'. Follow that path; as long as the path uses vertices and switches in H we can perform the same walk in H. Assume then that the path uses a switch of H' not in H. This switch must have been introduced to replace an ambiguous switch s in H. Let the neighbors of s in H be u, v, and w with $o(s) = u$. If the path uses the connection between u and v or u and w, then we can use s in H to traverse either usv or usw (both of which are acceptable). If, however, the path uses the connection between v and w in H', then in H we have to interpolate the walk from s to s (via $o(s) = u$). This results in an acceptable walk between v and w in H. Continuing this process, we obtain an acceptable walk in H from a to b.

To prove the reverse direction, suppose there is an acceptable walk from a to b in H. Then that same walk is an acceptable walk from a to b in H'. Hence, we can consider an acceptable walk W_{ab} from a to b in H' of minimal length. We will show that this walk is in fact a path.

Since W_{ab} is acceptable, all its interior nodes have to be switches. Suppose the walk contains a repetition of some switch s. Let s be the very first switch along W_{ab} which is repeated, and let it be connected to nodes u, v and w, and $o(s) = u$. If W_{ab} enters s along the same edge both times, we could simply eliminate the walk between s and s. The same is true, if the walk enters s once from v and once from w, since it has to continue to u in both cases. Since W_{ab} was chosen of minimal length, neither of these two cases could have occurred. Up to symmetry (between v and w) this leaves us with the following three cases:

(i) the path enters s from v the first time, and from u the second time, leaving along w,
(ii) the path enters s from v the first time, and from u the second time, leaving along v,
(iii) the path enters s from u the first time, and from v the second time.

Cases (ii) and (iii) cannot actually occur, since in both cases s would not have been the first switch along W_{ab} which is repeated (note that vertices cannot be repeated). Hence, we are in case (i). In this case, s has to be one of the three replacements of an ambiguous switch in H with v and w being the remaining two. However, this means in H' we could have directly proceeded from v to w, and would not have gone through s at all (since W_{ab} is of minimal length). Hence, none of the switches in W_{ab} can repeat, and it is indeed a path. ∎

Theorem 1 and Lemma 1 immediately imply 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 represented by the confluent drawing in Figure 5. (The even-indexed vertices induce a graph H in which 4 and 6 have neighbors $\{2, 8, 10, 12\}$ and $(10, 8, 2, 12)$ is an induced path subgraph.) By adding a vertex 0 adjacent to all odd-indexed vertices, we obtain a graph G in which the switches for $G - \{0\}$ must all be in (or out of) the circle. G is confluent but not strongly confluent, as can easily be checked.

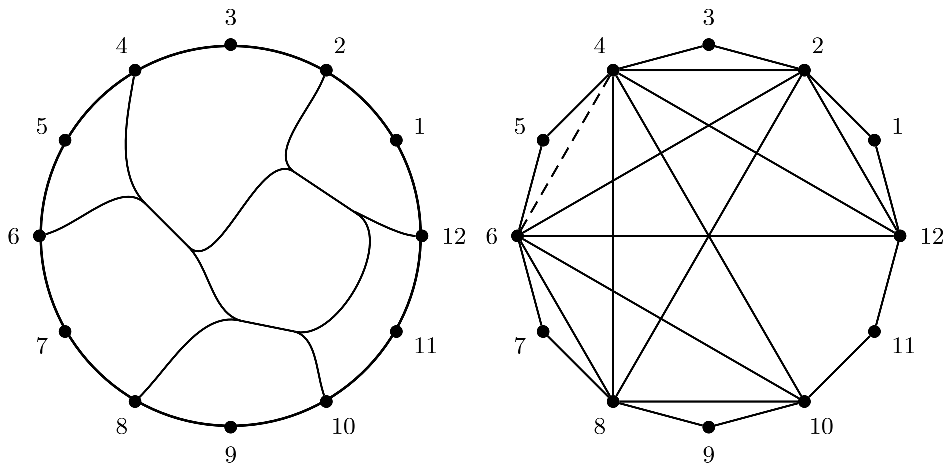

Fig. 5. Construction for graph which is confluent but not strongly confluent. *Left,* Train-track drawing of G; *right,* G (strongly confluent: with dashed edge; confluent: without).

4 Strong Confluency in NP

Theorem 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 ($e, f \in E$), 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

$$(o_{e_1f_1}, o_{e_2f_2}, \ldots, o_{e_kf_k})$$

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 Theorem 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 6.

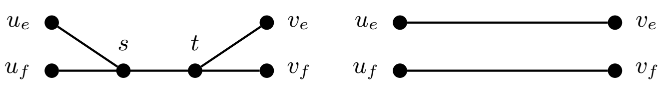

Fig. 6. Lifting a path (in a strongly confluent representation).

There is one scenario which would prohibit the application of the move shown in Figure 6, 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.

Remark 3. 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:

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

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

Proof. 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 $\overrightarrow{e}$ along P_{uv} from u to v. This yields a monotone function, namely, if $\overrightarrow{e}$ occurs before $\overrightarrow{f}$ 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 7 shows how the order of two parallel crossings can be swapped.

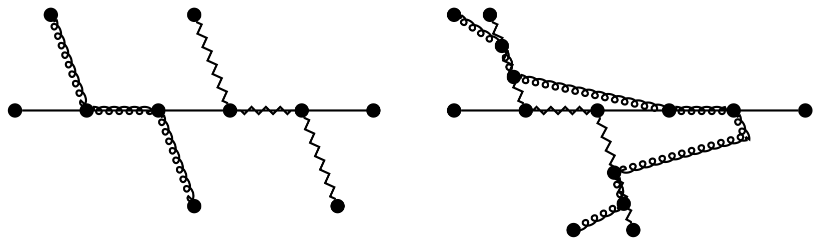

Fig. 7. How to swap two parallel crossings.

We can use a similar move for non-parallel crossings, as shown in Figure 8.

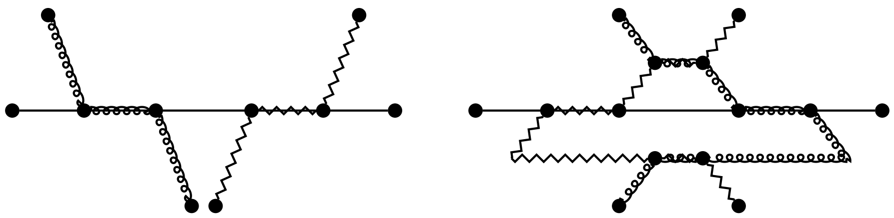

Fig. 8. How to swap two non-parallel crossings.

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. Lemma 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 (not necessarily smooth or locally monotone). For example, Figure 9 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*.[7] We will see later that all tree-confluent graphs are bipartite, so there is no tree-like train track drawing representing K_6.

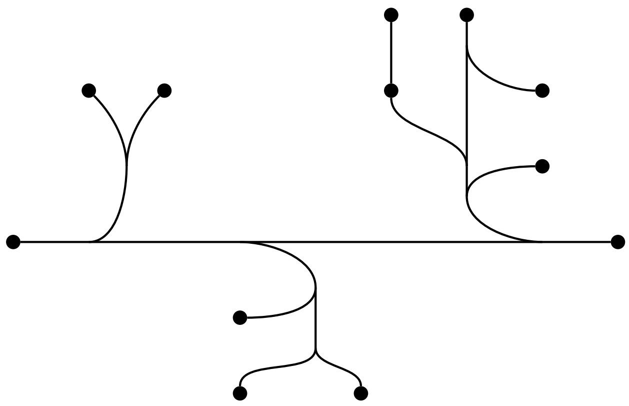

Fig. 9. A tree-like train track drawing.

In graph theoretic terms, the underlying train track of a tree-confluent graph has to be a tree.

Lemma 3. *A graph is tree-confluent if and only if it is represented by a train track that is a tree.*

[7] Since a tree-like train track drawing does not contain closed curves, the notions of confluency and strong confluency coincide, so we can use the term "represent" without ambiguity.

The proof of the lemma is immediate. We now give a characterization of tree-confluent graphs in terms of a vertex elimination ordering. This characterization will allow us to identify tree-confluent graphs as another well-known graph class.

Theorem 3. *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).

Proof. Call vertices of type (*i*) or (*ii*) *good*. Observe that if G is tree-confluent then it will still be tree-confluent after the removal of a good vertex. Furthermore, if G is not tree-confluent, it cannot become tree-confluent by removing a good vertex: 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 good vertices 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 a good vertex. As a matter of fact, we will show that there are at least two good vertices (unless G consists of a single vertex).

To that end, fix a tree-confluent drawing of G. By adding switches, if necessary, we may assume that each vertex of G has exactly one incident branch in the drawing. Then if T is the tree-like train track that represents G, its leaves S are precisely the vertices of G. Now either T is a star (in which case we are done) or $T - S$ has at least two leaves x, y; x and y each have two neighbors that are leaves in T. If either of x or y is a switch branching into two leaves of S, those two leaves are both removable. Otherwise, both x and y are adjacent to a vertex in S of degree one. In either case, we have at least two good vertices. ∎

We required vertices of type (*ii*) to have *nonempty* neighborhoods, obtaining graphs which are tree-like also in the sense of being connected. If we allow the removal of isolated vertices, the resulting graphs will be unions of tree-confluent graphs, or *forest-confluent graphs*.

In [9], Golumbic and Goss introduced the well-known class of graphs known as the *chordal bipartite graphs*, which are the bipartite graphs in which every cycle of length at least 6 contains a chord (that is, there are no induced cycles of length 6 or greater).

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. Hence every forest-confluent graph is chordal bipartite.

As a matter of fact, we can show more. A graph is $(6,2)$-*chordal* if every cycle of length at least 6 contains at least *two* chords [4].

Theorem 4. *The forest-confluent graphs are precisely the* $(6,2)$*-chordal bipartite graphs.*

This class of graphs has other well-known characterizations; for example, the $(6,2)$-chordal bipartite graphs are just the domino-free chordal bipartite graphs, where a *domino* is a bipartite graph consisting of a C_6 with a single chord.

A graph is *distance-hereditary* if the distance between two vertices does not change if any other vertex is removed (unless the distance becomes infinite). It turns out that the forest-confluent graphs are the bipartite distance-hereditary graphs.

This follows from a result of Bandelt and Mulder [1, Theorem 1] who showed that a graph is distance-hereditary if and only if it has an elimination ordering allowing the removal of vertices of degree 1 and the identification of vertices u and v for which $N(u) - \{v\} = N(v) - \{u\}$ (such vertices are called *twins*). Twins generalize our notion of two vertices having the same neighborhood by allowing an edge between the two twins (in which case the twins are called *true twins*). Since true twins are impossible in a bipartite graph, this shows the equivalence of being bipartite distance-hereditary and being forest-confluent. Together with another result from Bandelt and Mulder's paper [1, Theorem 2] which implies that the $(6,2)$-chordal bipartite graphs are the bipartite distance-hereditary graphs, this implies Theorem 4.

However, we decided to include a simpler, direct proof of Theorem 4 keeping the paper self-contained.

Recall that a vertex v is *good* if it is either of degree 1 or there is another vertex u such that $N(u) = N(v) \neq \emptyset$. We showed that every tree-confluent graph contains at least two good vertices unless it is trivial. Note that the same is true for forest-confluent graphs, since if the graph is disconnected, each component has at least one good vertex.

Any induced subgraph of a forest-confluent graphs is forest confluent (this follows from the elimination characterization). If G and H are forest-confluent graphs that share a single vertex, then their union is clearly forest-confluent as well.

For the proof of the Theorem 4 we make use of the following lemma.

Lemma 4 (Bondy, Murty [3, Exercise 2.2.12]). *If* $S_1, \ldots, S_n$ *are pairwise different subsets of* $\{1, \ldots, n\}$ *then there is an* x *in* $\{1, \ldots, n\}$ *such that the sets* $S_1 - \{x\}, \ldots, S_n - \{x\}$ *are still pairwise different.*

Proof. Draw an edge between two sets if they differ by exactly one element. Label each edge by the element on which the sets differ. Take a maximal spanning forest of the resulting graph. The forest has at most $n-1$ edges, hence there is at least one element x that is not mentioned. ■

Proof (of Theorem 4). We show that the forest-confluent graphs are precisely the domino-free chordal bipartite graphs, which are clearly equivalent to the $(6, 2)$-chordal bipartite graphs.

By the elimination characterization it is immediate that every forest-confluent graph is chordal bipartite and domino-free.

For the other direction, suppose that there exists a chordal bipartite, domino-free graph which is not forest-confluent. Let $G = (X \cup Y, E)$ be a minimal such graph. For any vertex v, $G - v$ is clearly chordal bipartite and domino-free, so $G - v$ must be forest-confluent. Hence v cannot be good, because then G would be forest-confluent as well.

Similarly, G must be 2-connected: If it were disconnected or had a cut-vertex, it would be the union of forest-confluent subgraphs such that G would be forest-confluent as well.

Claim. If G has two vertices u and v such that $d(u) = d(v) = 2$, then u and v cannot share a common neighbor.

Suppose u and v had a common neighbor x. Since there are no good vertices, $N(u) \neq N(v)$. Therefore in $G - x$ the shortest u, v-path has length at least four (there is such a path, since G is 2-connected). This gives us a cycle of length at least six in G. In this cycle all chords are incident to x (because u and v have degree 2, and we chose a shortest path from u to v). Consequently, there is a chord between x and every other vertex of the path from u to v resulting in an induced domino, which is a contradiction.

Since we know that $N(u) \neq N(v)$ for all u, v in Y (otherwise G would have a good vertex), by Lemma 4 there is an x in X such that $N(u) - x \neq N(v) - x$ for all u, v in Y. By minimality of G, $G - x$ is forest-confluent, so either $G - x$ is trivial (which is not possible) or $G - x$ contains two good vertices a, b. Then a, b each have degree one in $G - x$ (since we know that $N_{G-x}(a) \neq N_{G-x}(b)$). Since G has no good vertices, we must have $d_G(a) = d_G(b) = 2$. But then x is a common neighbor of a, b, which contradicts the claim. ■

6 Outerconfluent Graphs

In analogy with outerplanar graphs we can define (strongly) outerconfluent graphs to be those graphs which have (strongly) confluent drawings in which all vertices are on the boundary of the unbounded face. As in the case of confluency there are examples of graphs that are outerconfluent but not strongly outerconfluent; for an example, see the confluent drawing in Figure 5.

Dickerson, Eppstein, Goodrich, and Meng [5] showed—in effect—that all cographs are strongly outerconfluent, thereby also showing that strong outerconfluency is a strict superclass of tree-confluency. It does not seem unlikely that outerconfluent graphs can be recognized in polynomial time. While this remains an open problem, we can recognize some interesting subclasses of outerconfluent graphs.

Definition 5. *A bipartite graph $G = (X \,\dot{\cup}\, Y, E)$ is* outerconfluent bipartite *if it has an outerconfluent drawing in which the closure of the unbounded face can be partitioned into two connected regions, one containing X, and the other Y.*

Equivalently, there is an outerconfluent drawing of G in which X and Y are placed on two different parallel lines, and the drawing lies entirely between the two lines.

Remark 4. Any outerconfluent bipartite graph is outerconfluent and bipartite, but the reverse is not true: C_6 is outerconfluent and bipartite, but it cannot be drawn with the three vertices of the even partition on one line and the three remaining vertices on a parallel line, since every (nontrivial) switch would lead to a C_4.

It turns out that the outerconfluent bipartite graphs are just the bipartite permutation graphs; we will not define this graph class here, but give an equivalent characterization which is more useful for our proof.

Lemma 5 (Spinrad, Brandstädt, Stewart [12]). *A bipartite graph $G = (X \,\dot{\cup}\, Y, E)$ is a bipartite permutation graph if and only if it admits a strong ordering; that is, there are orderings of X and Y such that for all $s <_X s' \in X$, and $t <_Y t' \in Y$:*

$$st', s't \in E \Rightarrow st, s't' \in E.$$

Theorem 5. *The outerconfluent bipartite graphs are precisely the bipartite permutation graphs.*

Proof. We will show (i) that every bipartite permutation graph is outerconfluent bipartite (even in the strong sense), and (ii) that every outerconfluent bipartite graph is a bipartite permutation graph, thereby also proving that strongly outerconfluent bipartite and outerconfluent bipartite coincide.

To show inclusion (i) let $G = (X \,\dot{\cup}\, Y, E)$ be a bipartite permutation graph. Arrange the points of X and Y on two parallel lines in the order suggested by the strong ordering. Furthermore, include a line segment from $s \in X$ to $t \in Y$ if $st \in E$. By adjusting the points slightly, we can assume that no two line segments intersect in more than one point. If such an intersection point lies strictly between the two parallel lines, we replace it by two switches forking upwards and downwards towards the endpoints of the line segments. We claim that the resulting drawing is a strongly confluent drawing of G. Obviously, all edges of E are realized by the drawing; the question is, whether the drawing realizes any edge not in E. Let st be not in E, but realized by the drawing. Following the drawing from s to t, let st' be the the first line-segment traversed. Without loss of generality, we can assume that $t <_Y t'$. Similarly, let $s't$ be the last line-segment traversed when going from s to t. If $s <_X s'$, then Lemma 5 allows us to conclude that $st \in E$ (since $st' \in E$ and $ts' \in E$). Therefore, assume that $s' <_X s$. Now the connection from s to t has to cross the (imaginary) line from s to t moving downwards from the right of s to the left of t. Hence, there has to be some line-segment xy with $s' <_X s <_X x$ and $y <_Y t <_Y t'$. However, xy then intersects both st' and $s't$, allowing us to apply Lemma 5 repeatedly to infer the presence of both xt and sy, and, thence, st which we precluded by assumption.

For inclusion (ii) we assume we are given an outerconfluent bipartite drawing of a graph $G = (X \dot\cup Y, E)$ lying between two parallel lines containing sets X and Y. We claim that the ordering of the points on the lines is a strong ordering of G. Suppose that $s <_X s' \in X$, $t <_Y t' \in Y$, s connects to t', and s' connects to t. We can redraw the graph such that the connection from s to t' is a straight line. Now follow the connection from t to s'. It has to cross the line st' at some point using a switch. This switch cannot merge t and s towards t', since that would connect t to t', hence the switch merges t and t' towards s, and therefore connects t to s. Similarly, s' is connected to t', showing that G is a bipartite permutation graph with respect to the ordering of the points on the lines.

7 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 [11]).

Identifying large classes of confluent graphs remains a challenging task. We have suggested the notion of tree-confluency, and identified it as a well-known graph class which can be recognized in linear time, and for which there are other efficient algorithms (see [4]). The same result was obtained independently Goodrich, Eppstein, and Meng [7]. That paper also introduced a larger class of graphs called *delta-confluent* which are just the tree-confluent graphs with delta devices (see Figure 4). As it turns out, the delta-confluent graphs are precisely the distance-hereditary graphs.

We also suggest the further study of outerconfluent graphs. We showed that there is a natural subclass, the outerconfluent bipartite graphs, which coincides with the bipartite permutation graphs. This should only be a first step towards an understanding of outerconfluency.

References

1. Hans-Jürgen Bandelt and Henry Martyn Mulder. Distance-hereditary graphs. *J. Combin. Theory Ser. B*, 41(2):182–208, 1986.
2. Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall, 1999.
3. J. A. Bondy and U. S. R. Murty. *Graph theory with applications.* American Elsevier Publishing Co., Inc., New York, 1976.

4. Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph classes: a survey.* SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999.
5. 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.
6. David Eppstein, 2004. personal communication.
7. David Eppstein, Michael T. Goodrich, and Jeremy Yu Meng. Delta-confluent drawings. In *Proc. 13th Int. Symp. Graph Drawing (GD 2005)*, Lecture Notes in Computer Science. Springer-Verlag, 2005.
8. Michael R. Garey and David S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983.
9. M. Golombic and C. F. Goss. Perfect elimination and chordal bipartite graphs. *J. Graph Theory*, 2:155–163, 1978.
10. R. C. Penner and J. L. Harer. *Combinatorics of train tracks*, volume 125 of *Annals of mathematics studies.* Princeton University Press, 1992.
11. 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).
12. Jeremy Spinrad, Andreas Brandstädt, and Lorna Stewart. Bipartite permutation graphs. *Discrete Appl. Math.*, 18(3):279–292, 1987.