# On Strict (Outer-)Confluent Graphs

Henry Förster[1], Robert Ganian[2], Fabian Klute[2(✉)], and Martin Nöllenburg[2]

[1] University of Tübingen, Tübingen, Germany
foersth@informatik.uni-tuebingen.de
[2] Algorithms and Complexity Group, TU Wien, Vienna, Austria
{rganian,fklute,noellenburg}@ac.tuwien.ac.at

**Abstract.** A strict confluent (SC) graph drawing is a drawing of a graph with vertices as points in the plane, where vertex adjacencies are represented not by individual curves but rather by unique smooth paths through a planar system of junctions and arcs. If all vertices of the graph lie in the outer face of the drawing, the drawing is called a strict outer-confluent (SOC) drawing. SC and SOC graphs were first considered by Eppstein et al. in Graph Drawing 2013. Here, we establish several new relationships between the class of SC graphs and other graph classes, in particular string graphs and unit-interval graphs. Further, we extend earlier results about special bipartite graph classes to the notion of strict outerconfluency, show that SOC graphs have cop number two, and establish that tree-like ($\Delta$-)SOC graphs have bounded cliquewidth.

## 1 Introduction

Confluent drawings of graphs are geometric graph representations in the Euclidean plane, in which vertices are mapped to points, but edges are not drawn as individually distinguishable geometric objects. Instead, an edge between two vertices $u$ and $v$ is represented by a smooth path between the points of $u$ and $v$ through a crossing-free system of arcs and junctions. Since multiple edge representations may share some arcs and junctions of the drawing, this allows dense and non-planar graphs to be drawn in a plane way (e.g., see Fig. 2 for a confluent drawing of $K_5$). Hence confluent drawings can be seen as theoretical counterpart of heuristic edge bundling techniques, which are frequently used in network visualizations to reduce visual clutter in layouts of dense graphs [2,25].

More formally, a *confluent drawing* $D$ of a graph $G = (V, E)$ consists of a set of points representing the vertices of $G$, a set of junction points, and a set of smooth arcs, such that each arc starts and ends at either a vertex point or a junction, no two arcs intersect (except at common endpoints), and all arcs meeting in a junction share the same tangent line in the junction point. There

A poster containing some of the results of this paper was presented at GD 2017. Robert Ganian acknowledges support by the Austrian Science Fund (FWF, project P31336) and is also affiliated with FI MUNI, Brno, Czech Republic.


D. Archambault and C. D. Tóth (Eds.): GD 2019, LNCS 11904, pp. 147–161, 2019.
https://doi.org/10.1007/978-3-030-35802-0_12

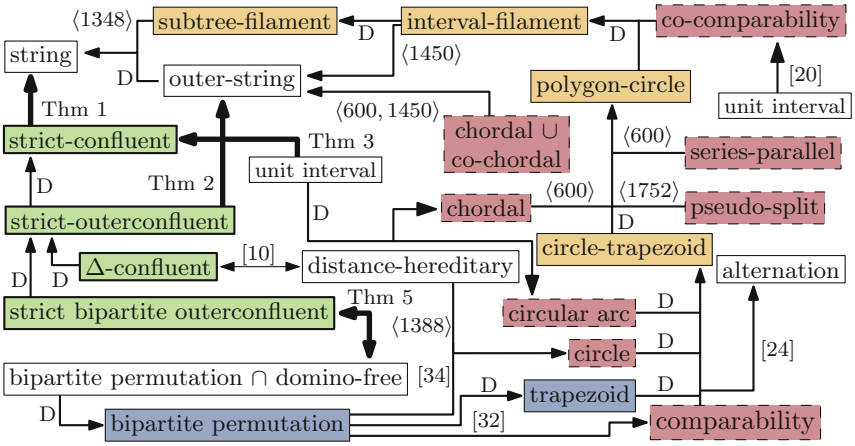


**Fig. 1.** Inclusions among graph classes related to SOC graphs. Arrows point from sub- to superclass, where edge label 'D' marks an inclusion by definition. Fat arrows are inclusions shown in this paper and are labelled with the corresponding theorem. Green boxes are confluent graph classes. Red, dashed boxes are classes that are incomparable to SOC graphs. Orange boxes are classes that are potential superclasses of SOC graphs. Blue boxes are potential subclasses of the SOC graphs. The numbers in $\langle\cdot\rangle$ indicate references of graphclasses.org. (Color figure online)

is an edge $(u, v) \in E$ if and only if there is a smooth path from $u$ to $v$ in $D$ not passing through any other vertex.

Confluent drawings were introduced by Dickerson et al. [8], who identified classes of graphs that admit or do not admit confluent drawings. Subsequently, the notions of strong and tree confluency have been introduced [27], as well as $\Delta$-confluency [10]. Confluent drawings have further been used for drawings of layered graphs [11] and Hasse diagrams [13].

Eppstein et al. [12] defined the class of strict confluent (SC) drawings, which require that every edge of the graph must be represented by a unique smooth path and that there are no self-loops. They showed that for general graphs it is NP-complete to decide whether an SC drawing exists. An SC drawing is called *strict outerconfluent* (SOC) if all vertices lie on the boundary of a (topological) disk that contains the SC drawing. For graphs with a given cyclic vertex order, Eppstein et al. [12] presented a constructive efficient algorithm for testing the existence of an SOC drawing. Without a given vertex order, neither the recognition complexity nor a characterization of such graphs is known.

We approach the characterization problem by comparing the SOC graph class with a hierarchy of classes of intersection graphs. In general a *geometric intersection graph* $G = (V, E)$ is a graph with a bijection between the vertices $V$ and a set of geometric objects such that two objects intersect if and only if the corresponding vertices are adjacent. Common examples include interval graphs, string graphs [9] and circle graphs [15]. Since confluent drawings make heavy use

of intersecting curves to represent edges in a planar way, it seems natural to ask what kind of geometric intersection models can represent a confluent graph.

**Contributions.** After introducing basic definitions and properties in Sect. 2, we show in Sect. 3 that SC and SOC graphs are, respectively, string and outerstring graphs [28]. Section 4 shows that every unit interval graph [30,33] can be drawn strict confluent. In Sect. 5, we consider the so-called strict bipartite-outerconfluent drawings: by following up on an earlier result of Hui et al. [27], we show that graphs which admit such a drawing are precisely the domino-free bipartite permutation graphs. Inspired by earlier work of Gavenčiak et al. [16], we examine in Sect. 6 the cop number of SOC graphs and show that it is at most two. In [14], we show additionally that many natural subclasses of outer-string graphs are incomparable to SOC graphs (see red, dashed boxes in Fig. 1). More specifically, we show that circle [15], circular-arc [26], series-parallel [31], chordal [17], co-chordal [4], and co-comparability [22] graphs are all incomparable to SOC graphs. This list may help future research by excluding a series of natural candidates for sub- and super-classes of SOC graphs. Finally, in Sect. 7, we show that the cliquewidth of so-called tree-like $\Delta$-SOC graphs is bounded by a constant, generalizing a previous result of Eppstein et al. [10].

*Due to space constraints some proofs are omitted; we refer to* [14] *for full details.*

## 2 Preliminaries

A *confluent diagram* $D = (N, J, \Gamma)$ in the plane $\mathbb{R}^2$ consists of a set $N$ of points called *nodes*, a set $J$ of points called *junctions* and a set $\Gamma$ of simple smooth curves called *arcs* whose endpoints are in $J \cup N$. Further, two arcs may only intersect at common endpoints. If they intersect in a junction they must share the same tangent line, see Fig. 2.

Let $D = (N, J, \Gamma)$ be a confluent diagram and let $u, v \in N$ be two nodes. A *uv-path* $p = (\gamma_0, \ldots, \gamma_k)$ in $D$ is a sequence of arcs $\gamma_0 = (u, j_1), \gamma_1 = (j_1, j_2), \ldots, \gamma_k = (j_k, v) \in \Gamma$ such that $j_1, \ldots j_k$ are junctions and $p$ is a smooth curve. In Fig. 2 the unique $uy$-path passes through junctions $i, j, k$. If there is at most one $uv$-path for each pair of nodes $u, v$ in $N$ and if there are no self-loops, i.e., no $uu$-path for any $u \in N$, we say that $D$ is a *strict* confluent diagram. The uniqueness of $uv$-paths and the absence of self-loops imply that every $uv$-path is actually a path in the graph-theoretic sense, where no vertex is visited twice. We further define $P(D)$ as the set of all smooth paths between all pairs of nodes in $N$. Let $p \in P(D)$ be a path and $j \in J$ a junction in $D$, then we write $j \in p$, if $p$ passes through $j$.

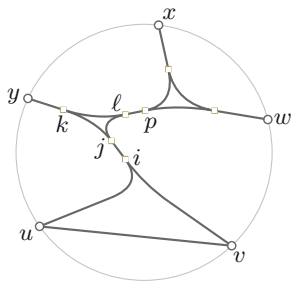


**Fig. 2.** A strict outerconfluent diagram representing $K_5$. Nodes are disks, junctions are squares.

As observed by Eppstein et al. [12], we may assume that every junction is a *binary* junction, where exactly three arcs meet such that the three enclosed

angles are $180°, 180°, 0°$. In other words two arcs from the same direction merge into the third arc, or, conversely, one arc splits into two arcs. A (strict) confluent diagram with higher-degree junctions can easily be transformed into an equivalent (strict) one with only binary junctions.

Let $j \in J$ be a binary junction with the three incident arcs $\gamma_1, \gamma_2, \gamma_3$. Let the angle enclosed by $\gamma_1$ and $\gamma_2$ be $0°$ and the angle enclosed by $\gamma_3$ and $\gamma_1$ (or $\gamma_2$) be $180°$. Then we say that $j$ is a *merge-junction* for $\gamma_1$ and $\gamma_2$ and a *split-junction* for $\gamma_3$. We also say that $\gamma_1$ and $\gamma_2$ *merge* at $j$ and that $\gamma_3$ *splits* at $j$. Given two nodes $u, v \in N$ and a junction $j \in J$ we say that $j$ is a merge-junction for $u$ and $v$ if there is a third node $w \in N$, a $uw$-path $p$ and a $vw$-path $q$ such that $j \in p$ and $j \in q$, the respective incoming arcs $\gamma_p = (j_p, j)$ and $\gamma_q = (j_q, j)$ are distinct and the suffix paths of $p$ and $q$ from $j$ to $w$ are equal. Conversely, we say that a junction $j \in J$ is a split-junction for a node $u \in N$ if there are two nodes $v, w \in N$, a $uv$-path $p$, and a $uw$-path $q$ such that $j \in p$ and $j \in q$, the prefix paths of $p$ and $q$ from $u$ to $j$ are equal and the respective subsequent arcs $\gamma_p = (j, j_p)$ and $\gamma_q = (j, j_q)$ are distinct. In Fig. 2, junction $i$ is a merge-junction for $u$ and $v$, while it is a split junction for each of $w, x, y$. Two junctions $i, j \in J$ are called a *merge-split pair* if $i$ and $j$ are connected by an arc $\gamma$ and both $i$ and $j$ are split-junctions for $\gamma$; in Fig. 2, junctions $i$ and $j$ form a merge-split pair, as well as junctions $\ell$ and $p$.

We call an arc $\gamma \in \Gamma$ *essential* if we cannot delete $\gamma$ without changing adjacencies in the represented graph. We call a confluent diagram $D$ *reduced*, if every arc is essential. Notice that this is a different notion than strictness, since it is possible that in a confluent diagram we find two essential arcs between a pair of nodes. Without loss of generality we can assume that the nodes of an outerconfluent diagram are placed on a circle with all arcs and junctions inside the circle. We can infer a *cyclic order* $\pi$ from an outerconfluent diagram $D$ by walking clockwise around the boundary of the unbounded face and adding the nodes to $\pi$ in the order they are visited.

From a confluent diagram $D = (N, J, \Gamma)$ we derive a simple, undirected graph $G_D = (V_D, E_D)$ with $V_D = N$ and $E_D = \{(u, v) \mid \exists\, uv\text{-path } p \in P(D)\}$. We say $D$ is a confluent drawing of a graph $G$ if $G$ is isomorphic to $G_D$ and that $G$ is a (strict) (outer-)confluent graph if it admits a (strict) (outer-)confluent drawing.

## 3 Strict (Outer-)Confluent $\subset$ (Outer-)String

The class of *string graphs* [28] contains all graphs $G = (V, E)$ which can be represented as the intersection graphs of open curves in the plane. We show that they form a superclass of SC graphs and that every SOC graph is an outer-string graph [28]. *Outer-string* graphs are string graphs that can be represented so that strings lie inside a disk and intersect the boundary of the disk in one endpoint. Note that strings are allowed to self-intersect and cross each more than once.

Let $D = (N, J, \Gamma)$ be a strict confluent diagram. For every node $u \in N$ we construct the *junction tree* $T_u$ of $u$, with root $u$ and a leaf for each neighbor $v$ of $u$ in $G_D$. The interior vertices of $T_u$ are the junctions which lie on the (unique) $uv$-paths. The strictness of $D$ implies that $T_u$ is a tree. Observe that every internal

node of $T_u$ has at most two children. Further, every merge-junction for $u$ is a vertex with one child in $T_u$, and every split-junction for $u$ has two children. For every junction $j$ in $T_u$ we can define the sub-tree $T_{u,j}$ of $T_u$ with root $j$.

**Lemma 1.** *Let $D = (N, J, \Gamma)$ be a strict confluent diagram, let $u, v \in N$ be two nodes and let $i, j$ be two distinct merge-junctions for $u, v$. Then $i$ is neither an ancestor nor a descendant of $j$ in $T_u$ (and, by symmetry, in $T_v$).*

To create a string representation of an SC graph we trace the paths of a strict confluent diagram $D = (N, J, \Gamma)$, starting from each node $u \in N$ and combine them into a string representation. Figure 3 shows an example. We traverse the junction tree for each $u \in N$ on the left-hand side of each arc (seen from its root $u$) and create a string $t(u)$, the *trace* of $u$, with respect to $T_u$ as follows.

Start from $u$ and traverse $T_u$ in left-first DFS order. Upon reaching a leaf $\ell$ make a clockwise U-turn and backtrack to the previous split-junction of $T_u$. When returning to a split-junction we have two cases. (a) coming from the left subtree: cross the arc from the left subtree at the junction and descend into the right subtree. (b) coming from the right subtree: cross the arc to the left subtree again and backtrack upward in the tree along the existing trace to the previous split-junction of $T_u$.

Finally, at a merge-junction $i$ with at least one trace from the other arc merging into $i$ already drawn: Let $v \in N$ such that $u$ and $v$ merge at $i$ and $t(v)$ is already tracing the subtree $T_{u,i} = T_{v,i}$.In this case we temporarily cut open the part of trace $t(v)$ closest to $t(u)$, route $t(u)$ through the gap and let it follow $t(v)$ along $T_{u,i}$ until it returns to junction $i$, where $t(u)$ passes through the gap again. Since $T_{u,i} = T_{v,i}$ this is possible without $t(u)$ intersecting $t(v)$. Now it remains to reconnect the two open ends of $t(v)$, but this can again be done without any new intersections by winding $t(v)$ along the "outside" of $t(u)$. See Fig. 3 for an illustration. If there are multiple traces with this property, they can all be treated as a single "bundled" trace within $T_{u,i}$.

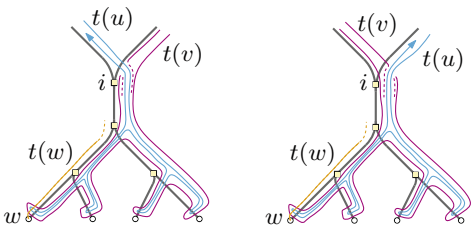


**Fig. 3.** Two possible configurations for inserting a new trace $t(u)$ that meets an existing trace $t(v)$ at a merge junction $i$; $t(v)$ is cut and re-routed.

**Theorem 1.** *Every SC graph is a string graph.*

*Proof.* Given an SC graph $G = (V, E)$ with a strict confluent drawing $D = (N, J, \Gamma)$ we construct the traces as described above for every node $u \in N$. In the following let $u, v$ be two nodes of $D$. We distinguish three cases.

**Case 1** ($uv$-path in $P(D)$): We draw $t(u)$ and $t(v)$ as described above. Since there is a $uv$-path in $P(D)$ we have to guarantee that $t(u)$ and $t(v)$ intersect at least once. We introduce crossings at the leaves corresponding to $u$ and $v$ in $T_u$

and $T_v$ when $t(u)$ and $t(v)$ make a U-turn; see how the trace $t(u)$ intersects $t(w)$ near the leaf $w$ in Fig. 3.

**Case 2** (No $uv$-path in $P(D)$ and $u, v$ share no merge-junction): In this case $T_u$ and $T_v$ are disjoint trees. Traces can meet only at shared junctions and around leaves, but since $t(u)$ and $t(v)$ trace disjoint trees intersections are impossible.

**Case 3** (No $uv$-path in $P(D)$ and $u, v$ share a merge-junction): First assume $u$ and $v$ share a single merge-junction $i \in J$ and assume $t(v)$ is already drawn when creating trace $t(u)$. We have to be careful that $t(v)$ and $t(u)$ do not intersect. If we route the traces at the merge-junction $i$ as depicted in Fig. 3, they visit the shared subtree $T_{u,i} = T_{v,i}$ without intersecting each other.

Now assume $u$ and $v$ share $k > 1$ merge-junctions $j_1, \ldots, j_k \in J$ and $u$ and $v$ merge at each $j_i$. Consequently we find $k$ shared subtrees $T^1, \ldots, T^k$. By Lemma 1, however, we know that the intersection of these subtrees is empty. Hence we can treat every merge-junction and its subtree independently as in the case of a single merge-junction.

These are all the cases how two junction trees can interact. Hence the traces $t(u)$ and $t(v)$ for nodes $u, v \in N$ intersect if and only if there is a $uv$-path in $P(D)$ and, equivalently, the edge $(u, v) \in E_D$. Further, every trace is a continuous curve, so this set of traces yields a string representation of $G$. □

A construction following the same principle can in fact be used to show:

**Theorem 2.** *Every SOC graph is an outer-string graph.*

## 4 Unit Interval Graphs and SC

In this section we consider so-called unit interval graphs. Let $G = (V, E)$ be a graph, then $G$ is a unit interval graph if there exists a *unit-interval* layout $\Gamma_{UI}$ of $G$, i.e. a representation of $G$ where each vertex $v \in V$ is represented as an interval of unit length and edges are given by the intersections of the intervals.

**Theorem 3.** *Every unit-interval graph is an SC graph.*

*Proof (Sketch).* Our proof technique is constructive and describes how to compute a strict confluent diagram $D$ for a given graph $G$ based on its unit-interval layout $\Gamma_{UI}$. Based on the ordering of intervals in $\Gamma_{UI}$, we first greedily compute a set of cliques which are subgraphs of $G$. In particular, we ensure that the left-to-right-ordered set of cliques has the property that vertices in a clique are only incident to vertices in the same clique and to the two neighboring cliques; see Fig. 4(a). We then create an SOC diagram for each clique; see the red, blue and green layouts of the three cliques in Fig. 4(b).

In order to realize the remaining edges we first make the following useful observation. Let $(v_1, \ldots, v_k)$ denote the vertices of some clique $C$ ordered from left to right according to $\Gamma_{UI}$. Then since all vertices are represented by unit intervals, if $v_i$ is incident to a vertex $w$ in the subsequent clique, also $v_j$ must be incident to $w$ for $i < j \leq k$. We use this observation to insert a split junction $b_i$

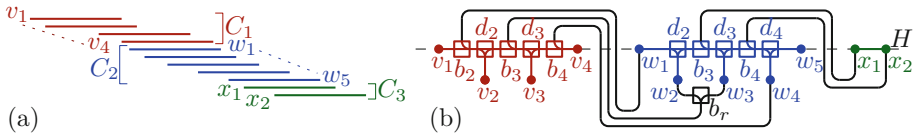


**Fig. 4.** (a) A unit interval graph $G$ with a decomposition of its vertices into a set of cliques as described in the proof of Theorem 3; and (b) a strict confluent layout of $G$ computed by the algorithm described in the proof of Theorem 3. (Color figure online)

in the SOC diagram of $C$ such that all vertices with index at least $i$ can access a smooth arc that connects them with $w$; see the black arcs in Fig. 4(b). We route arcs between cliques $C_i$ and $C_{i+1}$ first above clique $C_i$, then let it intersect with a line $H$ that passes through all the cliques (which intuitively inverts the ordering of such arcs) and then finish the drawing below clique $C_{i+1}$; refer to Fig. 4(b) for an illustration. By adopting this scheme for each pair of consecutive cliques, intersections can be prevented. □

## 5 Strict Bipartite-Outerconfluent Drawings

Let $G$ be a bipartite graph with bipartition $(X, Y)$. An outerconfluent drawing of $G$ is *bipartite-outerconfluent* if the vertices in $X$ (and hence also $Y$) occur consecutively on the boundary. Graphs admitting such a drawing are called *bipartite-outerconfluent*. The *bipartite permutation* graphs are just the graphs that are bipartite and *permutation* graphs, where a permutation graph is a graph that has an intersection model of straight lines between two parallel lines [29].

**Theorem 4 (Hui et al. [27]).** *The class of bipartite-permutation graphs is equal to the class of bipartite-outerconfluent graphs, i.e., the class of bipartite graphs admitting an intersection representation of straight-line segments between two parallel lines.*

It is natural to consider the idea of bipartite drawings also in the strict outerconfluent setting. We call a strict outerconfluent drawing $D$ of $G$ *bipartite* if it is bipartite-outerconfluent and strict. The graphs admitting such a drawing are called *strict bipartite-outerconfluent graphs*. In this section we extend Theorem 4 to the notion of strictness. The next lemma and observation are required in the proof of our theorem. The *domino* graph is the graph resulting from gluing two 4-cycles together at an edge.

**Lemma 2.** *Suppose that a reduced confluent diagram $D = (N, J, \Gamma)$ contains two distinct uv-paths. Then we can find in $G_D = (V_D, E_D)$ a set $V' \subseteq V_D$ such that $G[V']$ is isomorphic to $C_6$ with at least one chord.*

**Observation 1.** *Let $G = (V, E)$ be a graph and $V' \subseteq V$ a subset of six vertices such that $G[V']$ is isomorphic to a domino graph and let $X \cup Y = V'$ be the*

*corresponding bipartition. Now let $\pi$ be a cyclic order of $V'$ in which the vertices in $X$ and in $Y$ are contiguous, respectively. Then there is no strict outerconfluent diagram $D = (N, J, \Gamma)$ with order $\pi$ and $G_D = G[V']$ or, consequently, $G_D = G$.*

**Theorem 5.** *The (bipartite-permutation ∩ domino-free)-graphs are exactly the strict bipartite-outerconfluent graphs.*

*Proof (Sketch).* Let $G = (V, E)$ be a (bipartite-permutation ∩ domino-free) graph. By Theorem 4 we can find a bipartite-outerconfluent diagram $D = (N, J, \Gamma)$ which has $G_D = G$. Now assume that $D$ is reduced but not strict. In this case we find six nodes $N' \subseteq N$ corresponding to a vertex set $V' \subseteq V_D$ in $G_D$ such that $G_D[V'] = (V', E')$ is a $C_6$ with at least one chord by Lemma 2. In addition, since $D$ (and hence also $G_D$) is bipartite and domino-free, we know there are two or three chords. Then $G_D[V']$ is a $K_{3,3}$ minus one edge $e \in E'$ or $K_{3,3}$. In a bipartite diagram these can always be drawn in a strict way.

For the other direction, consider a strict bipartite-outerconfluent diagram $D = (N, J, \Gamma)$. By Theorem 4, $G_D$ is a bipartite permutation graph, and by Observation 1, it must be domino-free. Thus, $G_D$ must be as described. □

## 6 Strict Outerconfluent Graphs Have Cop Number Two

The *cops-and-robbers* game [1] on a graph $G = (V, E)$ is a two-player game with perfect information. The *cop-player* controls $k$ *cop tokens*, while the *robber-player* has one *robber token*. In the first move the cop-player places the cop tokens on vertices of the graph, and then the robber places his token on another vertex. In the following the players alternate, in each turn moving their tokens to a neighboring vertex or keeping them at the current location. The cop-player is allowed to move all cops at once and multiple cops may be at the same vertex. The goal of the cop-player is to catch the robber, i.e., place one of its tokens on the same vertex as the robber.

The *cop number* $cop(G)$ of a graph $G$ is the smallest integer $k$ such that the cop-player has a winning strategy using $k$ cop tokens. Gavenčiak et al. [16] showed that the cop number of outer-string graphs is between three and four, while the cop-number of many other interesting classes of intersection graphs, such as circle graphs and interval filament graphs, is two. We show that the cop number of SOC graphs is two as well.

Consider a SOC drawing $D = (N, J, \Gamma)$ of a graph $G = (V, E)$, which we can assume to be connected. For nodes $u, v \in N$, let the node interval $N[u, v] \subset N$ be the set of nodes in clockwise order between $u$ and $v$ on the outer face, excluding $u$ and $v$. Let the cops be located on nodes $C \subseteq N$ and the robber be located on $r \in N$. We say that the robber is *locked* to a set of nodes $N' \subset N$ if $r \in N'$ and every path from $r$ to $N \setminus N'$ contains at least one node that is either in $C$ or adjacent to a node in $C$; in other words, a robber is locked to $N'$ if it can be prevented from leaving $N'$ by a cop player who simply remains stationary unless the robber can be caught in a single move. The following lemma establishes that a single cop can lock the robber to one of two "sides" of a SOC drawing.

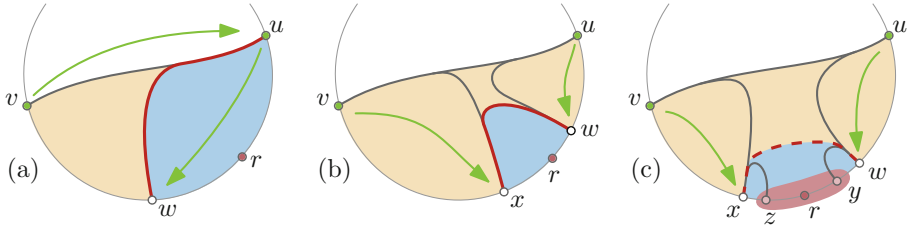


**Fig. 5.** Moves of the cops to confine the robber to a strictly smaller range.

**Lemma 3.** *Let $D = (N, J, \Gamma)$ be a SOC diagram of a graph $G$. Let a cop be placed on node $u$, the robber on node $r \neq u$ and not adjacent to $u$, and let $v \neq r$ be an arbitrary neighbor of $u$. Then the robber is either locked to $N[u, v]$ or locked to $N[v, u]$.*

Let $u, v \in N$ be two nodes of a SOC diagram $D = (N, J, \Gamma)$. We call a neighbor $w$ of $u$ in $N[u, v]$ *cw-extremal* (resp. *ccw-extremal*) for $u, v$ (assuming such a neighbor exists), if it is the last neighbor of $u$ in the clockwise (resp. counterclockwise) traversal of $N[u, v]$. Now let $u, v$ be two neighboring nodes in $N$, $w \in N[u, v]$ be the cw-extremal node for $u$ and $x \in N[u, v]$ be the ccw-extremal node for $v$. If $w$ appears before $x$ in the clockwise traversal of $N[u, v]$ we call $w, x$ the *extremal pair* of the $uv$-path, see Fig. 5(b) and (c). In the case where only one node of $u, v$ has an extremal neighbor $w$, say $u$, we define the extremal pair as $v, w$. In the following we assume that for a given $uv$-path the extremal pair exists.

**Lemma 4.** *Let $D = (N, J, \Gamma)$ be a SOC diagram of a graph $G$, $u, v \in N$ be two nodes connected by a $uv$-path in $P(D)$ and $w, x \in N[u, v]$ the extremal pair of the $uv$-path. If the cops are placed at $u$ and $v$ and the robber is at $r \in N[u, v]$, $r \neq w$, $r \neq x$, there is a move that locks the robber to $N[u, w]$, $N[w, x]$ or $N[x, v]$.*

**Lemma 5.** *Let $D = (N, J, \Gamma)$ be a SOC diagram of a graph $G$, $u, v \in N$ be two nodes connected by a $uv$-path in $P(D)$ and $w, x \in N[u, v]$ be the extremal pair of the $uv$-path such that there is no $wx$-path in $P(D)$. If the robber is at $r \in N[w, x]$ and the cops are placed on $w, x$ we can find $y, z \in N[w, x] \cup \{w, x\}$ such that the $yz$-path exists in $P(D)$ and the robber is locked to $N[y, z]$.*

Combining Lemmas 3, 4 and 5 yields the result.

**Theorem 6.** *SOC graphs have cop number two.*

*Proof (Sketch).* Let $D = (N, J, \Gamma)$ be a strict-outerconfluent diagram of a (connected) graph $G$. Choose any $uv$-path in $P(D)$ and place the cops on $u$ and $v$ as the initial move. The robber must be placed on a node $r$ that is either in $N[u, v]$ or in $N[v, u]$; by symmetry, let us assume the former. By Lemma 3, the robber is now locked to $N[u, v] \neq \emptyset$.

In every move we will shrink the locked interval until eventually the robber is caught. We distinguish three cases, based on the extremal neighbors $w$ and $x$ of $u$ and $v$ in $N[u, v]$ and their ordering along the outer face. If $w, x$ form no extremal pair, we can use Lemma 3, if they do form an extremal pair, we use first Lemma 4 and then, depending on the configuration, again Lemma 3 (see Fig. 5(b)) or go into the case of Lemma 5 (see Fig. 5(c)). □

Theorem 6 suggests a closer link between SOC graphs and interval-filament graphs [18], another subclass of outer-string graphs with cop number two.

## 7 Clique-Width of Tree-Like Strict Outerconfluent Graphs

In 2005, Eppstein et al. [10] showed that every strict confluent graph whose arcs in a strict confluent drawing topologically form a tree is distance hereditary and hence exhibits certain well-understood structural properties—in particular, every such graph has bounded *clique-width* [6]. These graphs are called *Δ-confluent* graphs. In their tree like confluent drawings an additional type of 3-way junction is allowed, the *Δ-junction*, which smoothly links together all three incident arcs. See Fig. 6, where the junctions $j'$ and $k'$ now form a single $\Delta$-junction instead of three separate merge or split junctions.

In this section, we lift the result of Eppstein et al. [10] to the class of strict outerconfluent graphs: in particular, we show that as long as the arcs incident to junctions (including $\Delta$-junctions) topologically form a tree, strict outerconfluent graphs also have bounded clique-width. Equivalently, we show that "extending" any drawing covered by Eppstein et al. [10] through the addition of outerplanar drawings of subgraphs in order to produce a strict outerconfluent drawing does not substantially increase the clique-width of the graph. Since the notion of clique-width will be central to this section, we formally introduce it below (see also the work of Courcelle et al. [6]). A $k$-graph is a graph whose vertices are labeled by $[k] = \{1, 2, \ldots, k\}$; formally, the graph is equipped with a labeling function $\gamma\colon V(G) \to [k]$, and we also use $\gamma^{-1}(i)$ to denote the set of vertices labeled $i$ for $i \in [k]$. We consider an arbitrary graph as a $k$-graph with all vertices labeled by 1. We call the $k$-graph consisting of exactly one vertex $v$ (say, labeled by $i$) an *initial* $k$-graph and denote it by $i(v)$. The clique-width of a graph $G$ is the smallest integer $k$ such that $G$ can be constructed from *initial* $k$-graphs by means of repeated application of the following three operations:

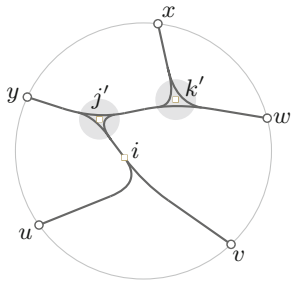


**Fig. 6.** A $\Delta$-confluent diagram representing $K_5 - (u, v)$. Nodes are disks, junctions are squares. $\Delta$-junctions are marked with a grey circle.

1. Disjoint union (denoted by $\oplus$);
2. Relabeling: changing all labels $i$ to $j$ (denoted by $p_{i\to j}$);
3. Edge insertion: adding an edge between every vertex labeled by $i$ and every vertex labeled by $j$, where $i \neq j$ (denoted by $\eta_{i,j}$ or $\eta_{j,i}$).

The construction sequence of a $k$-graph $G$ using the above operations can be represented by an algebraic term composed of $i(v)$, $\oplus$, $p_{i\to j}$ and $\eta_{i,j}$ (where $v \in V(G)$, $i \neq j$ and $i, j \in [k]$). Such a term is called a *$k$-expression* defining $G$, and the *clique-width* of $G$ is the smallest integer $k$ such that $G$ can be defined by a $k$-expression. Distance-hereditary graphs are known to have clique-width at most 3 [23] and outerplanar graphs have clique-width at most 5 due to having treewidth at most 2 [3,7].

Let *(tree-like) $\Delta$ -SOC graphs* be the class of all graphs which admit strict outerconfluent drawings (including $\Delta$-junctions) such that the union of all arcs incident to at least one junction topologically forms a tree. Clearly, the edge set $E$ of every tree-like $\Delta$-SOC graph $G = (V, E)$ with confluent diagram $D_G$ can be partitioned into sets $E_s$ and $E_c$, where $E_s$ (the set of *simple edges*) contains all edges represented by single-arc paths in $D$ not passing through any junction and $E_c$ (the set of confluent edges) contains all remaining edges in $G$. Let $G_c = G[E_c] = (V_c, E_c)$ be the subgraph of $G$ induced by $E_c$, i.e., $V_c$ is obtained from $V$ by removing all vertices without incident edges in $E_c$.

We note that even though $G_c$ is known to be distance-hereditary [10] and $G - E_c$ is easily seen to be outerplanar, this does not imply that tree-like $\Delta$-SOC graphs have bounded clique-width—indeed, the union of two graphs of bounded clique-width may have arbitrarily high clique-width (consider, e.g., the union of two sets of disjoint paths that create a square grid). Furthermore, one cannot easily adapt the proof of Eppstein et al. [10] to tree-like $\Delta$-SOC graphs, as that explicitly uses the structure of distance-hereditary graphs; note that there exist outerplanar graphs which are not distance-hereditary, and hence tree-like $\Delta$-SOC graphs are a strict superclass of distance hereditary graphs. Before proving the desired theorem, we introduce an observation which will later allow us to construct parts of $G$ in a modular manner.

**Observation 2.** *Let $H = (V, E)$ be a graph of clique-width $k \geq 2$, let $V_1, V_2$ be two disjoint subsets of $V$, and let $s \in V \setminus (V_1 \cup V_2)$. Then there exists a $(3k+1)$-expression defining $H$ so that in the final labeling all vertices in $V_1$ receive label 1, all vertices in $V_2$ receive label 2, $s$ receives label 3 and all remaining vertices receive label 4.*

**Theorem 7.** *Every tree-like $\Delta$-SOC graph has clique-width at most 16.*

*Proof (Sketch).* We begin by partitioning the edge set of the considered $\Delta$-SOC graph into $E_c$ and $E_s$, as explained above, and by setting an arbitrary arc incident to a junction as the root $r$. Given a tree-like $\Delta$-SOC drawing of the graph, our aim will be to pass through the confluent arcs of the drawing in a leaves-to-root manner so that at each step we construct a 16-expression for a certain circular segment of the outer face. This way, we will gradually build up

the 16-expression for $G$ from modular parts, and once we reach the root we will have a complete 16-expression for $G$.

At its core, the proof partitions nodes in the drawing into *regions*, delimited by arcs connecting nodes and junctions (such nodes are *not* part of any region). Each region is an outerplanar graph (which has clique-width at most 5), and furthermore the nodes in a region can only be adjacent to the nodes on the boundary of that region. Hence, by Observation 2 using $k = 5$, each region can be constructed by a 16-expression which also uses separate labels to capture the neighborhood of that region to its border. See Fig. 7 for an illustration.

The second ingredient used in the proof is tied to the tree-like structure of the drawing. In particular, one cannot construct a 16-expression (and even any $k$-expression for constant $k$) by simply joining the regions together in the order they appear along the outer face. Instead, to handle the adjacencies imposed by the paths in the drawing, one needs to process regions (and their bordering vertices) in an order which respects the structure of the tree. To do so, we introduce a notion of *depth*: nodes have a depth of 0, while junctions have depth equal to the largest depth of its "children" plus 1. Regions are then processed in an order which matches the depth of the corresponding junctions: for instance, if in Fig. 7 one of the junctions $a_1$ and $a_2$ has depth $d$ then junction $j'$ has depth $d + 1$, and so the blue regions will be constructed by modular 16-expressions before the yellow one. Afterwards, all three regions $R_1, R_2, R_3$ will be merged together into a blue region with a single 16-expression. By iterating this process, upon reaching the root $r$ we obtain a 16-expression that constructs the whole $\Delta$-SOC graph. □

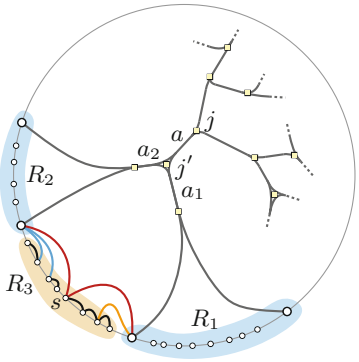


**Fig. 7.** Sketch of a tree-like $\Delta$-SOC graph $G$ with its regions. (Color figure online)

## 8 Conclusion

While this work provides the first in-depth study of SC and SOC graphs, a number of interesting open questions remain. One such question is motivated by our results on the cop-number of SOC graphs: we showed that SOC graphs are incomparable to most classes identified to have cop number two by Gavenčiak et al. [16], but we could not show such a result for the class of interval-filament graphs [18]. It seems likely that SOC graphs are contained in this class. Similarly, it is open whether SC graphs are contained in subtree-filament graphs. Furthermore, it is conceivable that a similar construction for the inclusion in string graphs, Sect. 3, could be used to show similar results for non-strict confluent graphs. Finally, investigating the curve complexity of our construction might provide insight into the curve complexity of SC and SOC diagrams.

On the algorithmic side, Sect. 7 raises the question of whether clique-width might be used to recognize SOC graphs, and perhaps even for finding SOC drawings. Another decomposition-based approach would be to use so-called split-decompositions [19], which we did not consider here. It is also open whether all bipartite permutation and trapezoid graphs [5,21] are SOC graphs. Since bipartite permutation graphs are equivalent to bipartite trapezoid graphs [5,21], the former represents a promising first step in this direction. It also remains open if it is possible to drop the unit length condition on the intervals in Sect. 4. We did not see an obvious way of adapting the construction for confluent drawings of interval graphs [8].

## References


1. Aigner, M., Fromme, M.: A game of cops and robbers. Discrete Appl. Math. **8**(1), 1–12 (1984). https://doi.org/10.1016/0166-218X(84)90073-8
2. Bach, B., Riche, N.H., Hurter, C., Marriott, K., Dwyer, T.: Towards unambiguous edge bundling: investigating confluent drawings for network visualization. IEEE Trans. Vis. Comput. Graph. **23**(1), 541–550 (2017). https://doi.org/10.1109/TVCG.2016.2598958
3. Baker, B.S.: Approximation algorithms for NP-complete problems on planar graphs. J. ACM **41**(1), 153–180 (1994). https://doi.org/10.1145/174644.174650
4. Benzaken, C., Crama, Y., Duchet, P., Hammer, P.L., Maffray, F.: More characterizations of triangulated graphs. J. Graph Theory **14**(4), 413–422 (1990). https://doi.org/10.1002/jgt.3190140404
5. Brandstädt, A., Spinrad, J., Stewart, L.: Bipartite permutation graphs are bipartite tolerance graphs. Congressus Numerantium **58**, 165–174 (1987)
6. Courcelle, B., Makowsky, J.A., Rotics, U.: Linear time solvable optimization problems on graphs of bounded clique-width. Theory Comput. Syst. **33**(2), 125–150 (2000). https://doi.org/10.1007/s002249910009
7. Courcelle, B., Olariu, S.: Upper bounds to the clique width of graphs. Discrete Appl. Math. **101**(1–3), 77–114 (2000). https://doi.org/10.1016/S0166-218X(99)00184-5
8. Dickerson, M., Eppstein, D., Goodrich, M.T., Meng, J.Y.: Confluent drawings: visualizing non-planar diagrams in a planar way. J. Graph Algorithms Appl. **9**(1), 31–52 (2005). https://doi.org/10.7155/jgaa.00099
9. Ehrlich, G., Even, S., Tarjan, R.E.: Intersection graphs of curves in the plane. J. Comb. Theory Ser. B **21**(1), 8–20 (1976). https://doi.org/10.1016/0095-8956(76)90022-8
10. Eppstein, D., Goodrich, M.T., Meng, J.Y.: Delta-confluent drawings. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS, vol. 3843, pp. 165–176. Springer, Heidelberg (2006). https://doi.org/10.1007/11618058_16
11. Eppstein, D., Goodrich, M.T., Meng, J.Y.: Confluent layered drawings. Algorithmica **47**, 439–452 (2007). https://doi.org/10.1007/s00453-006-0159-8
12. Eppstein, D., Holten, D., Löffler, M., Nöllenburg, M., Speckmann, B., Verbeek, K.: Strict confluent drawing. J. Comput. Geom. **7**(1), 22–46 (2016). https://doi.org/10.20382/jocg.v7i1a2
13. Eppstein, D., Simons, J.A.: Confluent Hasse diagrams. J. Graph Algorithms Appl. **17**(7), 689–710 (2013). https://doi.org/10.1007/978-3-642-25878-7_2

14. Förster, H., Ganian, R., Klute, F., Nöllenburg, M.: On strict (outer-)confluent graphs. CoRR abs/1908.05345 (2019). http://arxiv.org/abs/1908.05345
15. Gabor, C.P., Supowit, K.J., Hsu, W.L.: Recognizing circle graphs in polynomial time. J. ACM **36**(3), 435–473 (1989). https://doi.org/10.1145/65950.65951
16. Gavenčiak, T., Jelínek, V., Klavík, P., Kratochvíl, J.: Cops and robbers on intersection graphs. In: Cai, L., Cheng, S.-W., Lam, T.-W. (eds.) ISAAC 2013. LNCS, vol. 8283, pp. 174–184. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45030-3_17, https://doi.org/10.1016/j.ejc.2018.04.009
17. Gavril, F.: Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. SIAM J. Comput. **1**(2), 180–187 (1972). https://doi.org/10.1137/0201013
18. Gavril, F.: Maximum weight independent sets and cliques in intersection graphs of filaments. Inf. Process. Lett. **73**(5–6), 181–188 (2000). https://doi.org/10.1016/S0020-0190(00)00025-9
19. Gioan, E., Paul, C.: Split decomposition and graph-labelled trees: characterizations and fully dynamic algorithms for totally decomposable graphs. Discrete Appl. Math. **160**(6), 708–733 (2012). https://doi.org/10.1016/j.dam.2011.05.007
20. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs, vol. 57. Elsevier, Amsterdam (2004). https://doi.org/10.1002/net.3230130214
21. Golumbic, M.C., Monma, C.L., Trotter Jr., W.T.: Tolerance graphs. Discrete Appl. Math. **9**(2), 157–170 (1984). https://doi.org/10.1016/0166-218X(84)90016-7
22. Golumbic, M.C., Rotem, D., Urrutia, J.: Comparability graphs and intersection graphs. Discrete Math. **43**(1), 37–46 (1983). https://doi.org/10.1016/0012-365X(83)90019-5
23. Golumbic, M.C., Rotics, U.: On the clique-width of some perfect graph classes. Int. J. Found. Comput. Sci. **11**(3), 423–443 (2000). https://doi.org/10.1142/S0129054100000260
24. Halldórsson, M.M., Kitaev, S., Pyatkin, A.: Alternation graphs. In: Kolman, P., Kratochvíl, J. (eds.) WG 2011. LNCS, vol. 6986, pp. 191–202. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25870-1_18
25. Holten, D.: Hierarchical edge bundles: visualization of adjacency relations in hierarchical data. IEEE Trans. Vis. Comput. Graph. **12**(5), 741–748 (2006). https://doi.org/10.1109/TVCG.2006.147
26. Hsu, W.L.: Maximum weight clique algorithms for circular-arc graphs and circle graphs. SIAM J. Comput. **14**(1), 224–231 (1985). https://doi.org/10.1137/0214018
27. Hui, P., Pelsmajer, M.J., Schaefer, M., Stefankovic, D.: Train tracks and confluent drawings. Algorithmica **47**(4), 465–479 (2007). https://doi.org/10.1007/s00453-006-0165-x
28. Kratochvíl, J.: String graphs. I. The number of critical nonstring graphs is infinite. J. Comb. Theory Ser. B **52**(1), 53–66 (1991). https://doi.org/10.1016/0095-8956(91)90090-7
29. Pnueli, A., Lempel, A., Even, S.: Transitive orientation of graphs and identification of permutation graphs. Can. J. Math. **23**(1), 160–175 (1971). https://doi.org/10.4153/CJM-1971-016-5
30. Roberts, F.S.: Indifference graphs. In: Proof Techniques in Graph Theory, pp. 139–146 (1969)
31. Takamizawa, K., Nishizeki, T., Saito, N.: Linear-time computability of combinatorial problems on series-parallel graphs. J. ACM **29**(3), 623–641 (1982). https://doi.org/10.1145/322326.322328
32. Trotter, W.T.: Combinatorics and Partially Ordered Sets: Dimension Theory, vol. 6. JHU Press, Baltimore (2001). https://doi.org/10.1137/1035116

33. Wegner, G.: Eigenschaften der Nerven homologisch-einfacher Familien im Rn. Ph.D. thesis, Universität Göttingen (1967)
34. Yu, C.W., Chen, G.H.: Efficient parallel algorithms for doubly convex-bipartite graphs. Theoret. Comput. Sci. **147**(1–2), 249–265 (1995). https://doi.org/10.1016/0304-3975(94)00220-D