Strict confluent drawing

Soeren Nickel
Matr.: 01528974
Curriculum: 186.862 - Seminar in Algorithms Graphs and Geometry
e01528974@student.tuwien.ac.at

December 24, 2017

Abstract

This paper aims to present strict confluent drawings, a way of visualizing non-planar Graphs in a crossing free way, and tries to place them in a current field of research. Adjacencies are modeled through a smooth path between two vertices, which can merge and split at junctions. An additional restriction (strictness) assures unique paths and no self loops.

1 Introduction

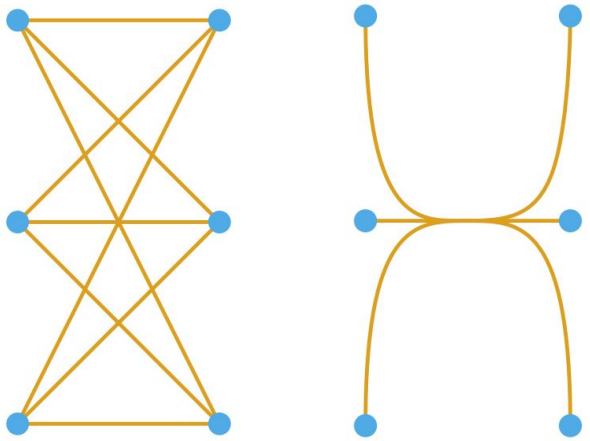
Strict Confluent Drawings are a subclass of confluent drawings.In a confluent drawing, the vertices of the graph are the vertices of the drawing. Adjacencies between two vertices are represented by a smooth path from one vertex to the other. This path is composed of arcs and junctions between arcs. every arc is adjacent to either a junction or a vertex at both ends. The resulting representation resembles a system of train tracks that merge and split at turnouts. Two arcs that meet at such a junction have, at the point of the junction, the same tangent. This leads to the possibility to enter the junction on one side and leave it on the other all on a smooth path, but makes a jump from one arc to another one, which enters the junction from the same side, impossible. A path from one vertex to another, that represents an adjacency, does not pass through another vertex.

Graph drawings can be optimized for better readability by optimizing various different characteristics and aesthetics of the drawing [17]. One of these aesthetics is to minimize the number of edge-crossings in a drawing. With confluent drawings we have the ability to visualize even very dense non-planar graphs in a visually improved manner. As we can see in figure 1 on page 2. Confluent drawings can share a visual similarity to another method of graph drawing called edge bundling [2, 4, 9, 10, 12], in which edges are contracted to be near each other when they cross other edges in order to cluster multiple crossings in similar directions into something that might be recognized as one big crossing. However in contrast to edge bundling, confluent drawings have the advantage that they do not create false adjacencies, are unambiguous and stay "information faithful" [13].

The main contribution [7] which we will look at in this paper is about strict confluent drawings. The "strictness" adds the following restrictions to a drawing of a graph:

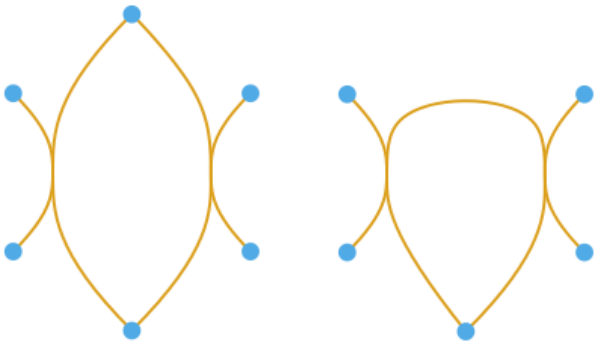
- If there exists a smooth path between two vertices, this path is unique. There cannot be another smooth path connecting these two vertices.
- There cannot be a smooth path from a vertex to itself

Through these restrictions, two things are intended to be achieved. First a better readability of a drawing since every path is unique. Secondly, by restricting the graph class like this, it is possible to completely characterize the complexity of the drawings, which is "the first such characterization for any form of confluence on arbitrary undirected graphs" [7]. The restrictions are visualized in figure 2 on page 2. We will talk in depth about the complexity of the drawings, as well as the complexity of recognizing if a given graph has a strict confluent drawing or not, in chapters 4 and 5.



(a) node link (b) confluent

Figure 1: A complete Graph $K_{3,3}$ as a classic node-link drawing (a) and as a confluent drawing (b)



(a) not unique path (b) self loop

Figure 2: Graph drawings that violate either uniqueness of paths (a) or prohibition of self loops (b)

2 Related Work

Confluent Drawings were introduced by Dickerson et al. [3], who provide 2 heuristic algorithms (for directed and undirected graphs), focusing on graphs with bounded arboricity. They also identify large graph classes (e.g. complement of trees, interval) which can be realized as a confluent drawing as well as a proof that there exist graphs for which such a drawing does not exist (e.g. Petersen graph, 4-dimensional hypercube). Additional work on confluent drawings include work on tree confluent drawings [11], where the confluent drawing of a graph is tree-like and an extension of tree confluent drawings called Δ-confluent drawings [5], where the junctions are allowed to join up to 3 arcs at the same time. These Drawings can be drawn in $\mathcal{O}(n \log n)$ area on a hexagonal grid according to Eppstein et al. [5]. Strong confluent drawings which are a subclass of confluent drawings can be recognized in polynomial time. Another extension are the so called layered confluent drawings [6]. Here the concept of layered drawings is combined with confluence by connecting adjacent layers with confluent bicliques.

There are a couple of heuristic algorithms for creating confluent drawings. One method uses the biclique edge cover graph of a graph to create a confluent drawing [8], another one is based on rectangular duals [18]. Bach et al. [1] present an algorithm to construct confluent drawings from arbitrary directed and undirected graphs. They also compare confluent drawings with other graph drawing methods and found that unexperienced users "without particular expertise in visualization or network analysis are able to read small CDs without difficulty" and that "compared to existing bundling techniques, CDs are more likely to allow people to correctly perceive connectivity".

Even though confluent drawings are not yet used in real world applications [1, 13], there is potential for them to be used since a multitude of problems and applications contain non-planar graphs which can be realized in a readable way through a confluent way. One such example is the (partial) visualization of real-world multivariate datasets, that are too large to display in their entirety [13].

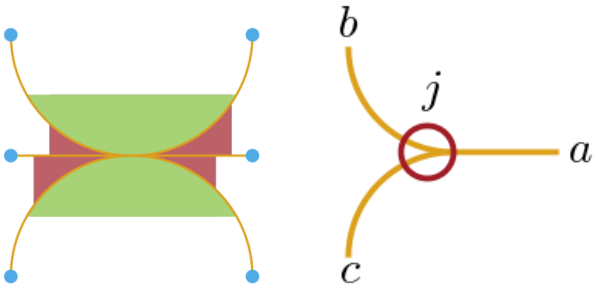
The restriction to strict confluent drawings, as mentioned above, did not see much work yet. The introductory paper [7] which is the main focal point of this paper as well as the report from Dagstuhl Seminar 13151 [14], from which it originated, are at the point of writing the only contribution to this topic. They introduce three different results:

- It is NP-complete to determine whether a given graph has a strict confluent drawing. To prove this, they reduce from planar 3-SAT and they use the fact that K_4 has exactly two strict confluent drawings.

- For a given graph, with a given cyclic ordering of its n vertices, there is an $\mathcal{O}(n^2)$-time algorithm to find an outer planar strict confluent drawing, if it exists: this is a drawing in a (topological) disk, with the vertices in the given order on the boundary of the disk. They define a canonical diagram which is a unique representation of all possible outer planar strict confluent drawings, where some of the faces are marked as cliques. They compute a canonical diagram by first computing its junctions, and then computing the arcs between them. From a canonical diagram one can easily derive an outer planar strict confluent drawing.

- When a graph has an outer planar strict confluent drawing, an algorithm based on circle packing can construct a layout of the drawing in which every arc is drawn using at most two circular arcs

Outerplanarity, as mentioned in the second and last bullet point, refers to a representation of graph in which all vertices can be aligned on the boundary of a circle and every edge between two vertices on that circle is contained inside the circle.



(a) Sharp angles (b) Merge and Split

Figure 3: Left: Sharp (green) and non sharp (red) angles at a junction
Right: Junction j is a split junction for arc a and conversely a merge junction for arc b and c

3 Preliminaries

For the sake of consistency we will not deviate from the preliminary definitions made by Eppstein et al. [7]. We call an edge e in a drawing D direct if it consists only of a single arc (that does not pass through junctions). We call the angle between two consecutive arcs at a junction or vertex sharp if the two arcs do not form a smooth path; each junction has exactly two angles that are not sharp, and every angle at a vertex is sharp (so the number of sharp angles at that vertex equals the degree of the vertex). Figure 3a on page 4 is an example of sharp and non sharp edges at junctions.

Let j be a junction. Then the two non-sharp angles at j separate the incident arcs into two disjoint sets A and B. Let a be an arc in A and $|B| > 1$. Then we say that j is a split junction for a since the path entering j via a splits at j into two or more paths. Conversely, if $|A| > 1$, we say that j is a merge junction for a since at least one more path merges with a at j. Figure 3b on page 4 shows the two cases.

Lemma 3.1. *Let $G = (V, E)$ be a graph, and let $E_0 \subseteq E$ be the edges of E that are incident to at least one vertex of degree 2. If G has a strict confluent drawing D, then it also has a strict confluent drawing D_0 in which all edges in E_0 are direct.*

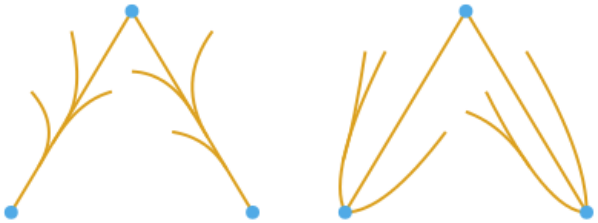
Proof. Let v be a degree-2 vertex in G with two incident edges e and f. There are two cases.

- If e and f leave v on two disjoint paths, then these paths have only merge junctions from vs perspective. Otherwise there would be a third edge that is incident to v. We can simply separate these junctions from e and f as shown in figure 4 on page 5.
- If on the other hand, e and f share the same path leaving v, then their paths split at some point. We need to reroute the merge junctions prior to the split and separate the merge junctions after the split as shown in Fig. 5. This is always possible since v has no other incident edges. Because D was strict and these changes do not affect strictness, D_0 is still a strict confluent drawing and edges e and f are direct.

□

Lemma 3.2. *Let G be a graph. If G has no $K_{2,2}$ as a subgraph, whose vertices all have degrees ≤ 3 in G, then G has a strict confluent drawing if and only if G is planar.*

Proof. Since every planar drawing is also a strict confluent drawing, that implication is obvious. So let D be a strict confluent drawing for a graph G without a $K_{2,2}$ subgraph, whose vertices all have degrees ≤ 3 in G. Since large junctions, where more than three arcs meet, can easily be transformed into an equivalent sequence of binary junctions, we can assume that every junction in D is binary, i.e., two arcs merge into one (or, from a different perspective, one arc splits into two). By Lemma 1 we can further transform D so that all edges incident to degree-2 vertices are direct. Now assume that



(a) Original (b) Transformed

Figure 4: First Case

(a) Original (b) Transformed

Figure 5: Second Case

there is a vertex u that has a path to a vertex v in D along which it visits a merge junction before visiting a split junction. Since D is strict, this situation implies the existence of a $K_{2,2}$ subgraph. Without the strict criteria, both paths could lead to the same vertex, in a strict drawing, both need to leave to a different vertex. All four vertices of that $K_{2,2}$ subgraph must have degree at least 3 as we have transformed all edges incident to degree-2 vertices into direct edges that do not pass through any junctions. This is a contradiction to the fact that G has no such subgraph. So the sequence of junctions on any $u-v$ path in D consists of a number of split junctions followed by a number of merge junctions. But any such path can be unbundled from its junctions to the left and right and turned into a direct edge without creating arc intersections. Repeating he argument shows that D can be transformed into a standard planar drawing of G. □

4 Combinatorial Complexity

This section will only be talking about strict confluent drawings with binary junctions. Let G be graph with a given confluent drawing. Note that it's always possible to transform this drawing into a drawing where every junction has degree three and every vertex has degree one (that is one adjacent arc, not one adjacency to another vertex). This transformation maintains the number of faces and can only increase the number of arcs and junctions. Consider the following two points.

Lemma 4.1. *A strict confluent drawing D cannot contain any smooth closed curves (circles).*

Proof. this would lead to either multiple or irrelevant paths, since either:

- only one arc is connected to the circle, in which case the circle is not part of a path from one vertex to another one and is therefore irrelevant in this drawing, or
- more than one arcs are connected to the circle, in which case some part of the circle is part of a path from one vertex to another. On this path we can always at the junction where the path leaves the circle decide not to leave the circle and instead go a full round around the circle. at the end of that round we will end up at the same junction again and we can still reach the end vertex of the path, which implies the existence of non unique paths.

□

Lemma 4.2. *Every face in a strict confluent drawing D must have at least three sharp angles.*

Proof. If a face f is adjacent to no sharp angle, then the boundary must form a smooth curve which implies the existence of a circle (contradiction to 4.1). If it is adjacent to exactly one sharp angle, there must be a path from a vertex back to itself which trails exactly around the face f. If a face f would be adjacent to exactly two sharp angles, we could identify at least one vertex which is either itself adjacent to the face (and therefore the reason for the sharp angle) or is reachable if we follow any path from the boundary of f through the junction. Since we do not have loops, we need to eventually reach a vertex. The same can be done for the other sharp angle. When we have identified those to vertices the path from one to the other must pass through both junctions. Since the two junctions are connected by exactly two trails (one above and one below f) the two vertices are connected by more than one path, which cannot happen. □

Let in the following:

- n be the number of vertices
- k be the number of junctions
- m be the number of arcs
- F be the number of faces
- c be the number of sharp angles

Lemma 4.3. *Every strict confluent drawing has at most* $2n-4$ *faces,* $5n-12$ *junctions, and* $8n-18$ *arcs.*

Proof. By double counting we get that $2m = n+3k$. The total number of sharp angles is $c = n+k$, and Lemma 4.2 implies that $c \geq 3F$, so that $n + k \geq 3F$. By combining the above relations with Euler's formula $n + k - m + F = 2$, we directly obtain that $F \leq 2n - 4$, $k \leq 5n - 12$, and $m \leq 8n - 18$. □

For outerplanar strict confluent drawings the following bounds can be given.

Lemma 4.4. *Every outer planar strict confluent drawing has at most n - 2 internal faces, 3n - 6 junctions, and 5n - 9 arcs.*

Proof. By double counting we get that $2m = n + 3k$. The total number of sharp angles in internal faces is $c = k$ (every vertex is on the outer face), and Lemma 4.2 implies that $c \geq 3F_{in}$, so that $k \geq 3F_{in}$. By combining the above relations with Euler's formula $n + k - m + F_{in} = 1$, we directly obtain that $F_{in} \leq n - 2$, $k \leq 3n - 6$, and $m \leq 5n - 9$. □

5 Computational Complexity

This is the main part of this paper. It is NP-Complete to decide whether a graph G has a strict confluent drawing in which all edges incident to a vertex with degree 2 are direct, which by 3.1 is enough to show that it is also complete to decide whether G has any strict confluent drawing. This will be shown by reduction from planar 3-SAT [15].

Remark: It is important to emphasize that (similar to section 3) we aim to explain the proof, presented by Eppstein et al. [7] in our own terms. The goal is to provide another approach to an interesting and well thought out proof, in order to further understanding, not only of the proof but the general procedure of it, since this kind of proof is often encountered in the field of algorithmic geometry.

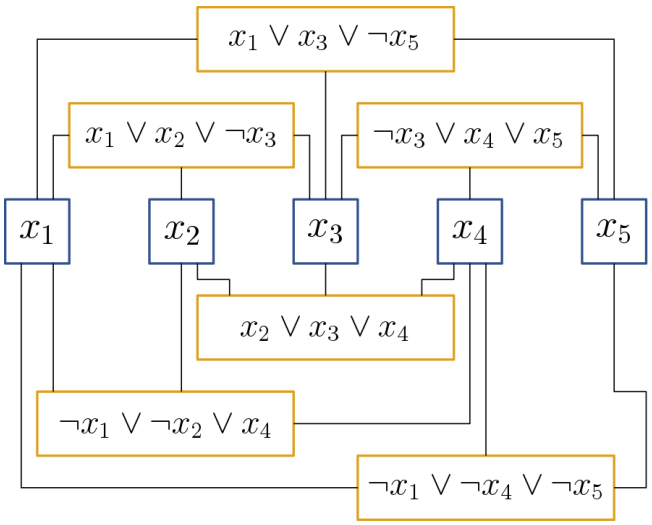
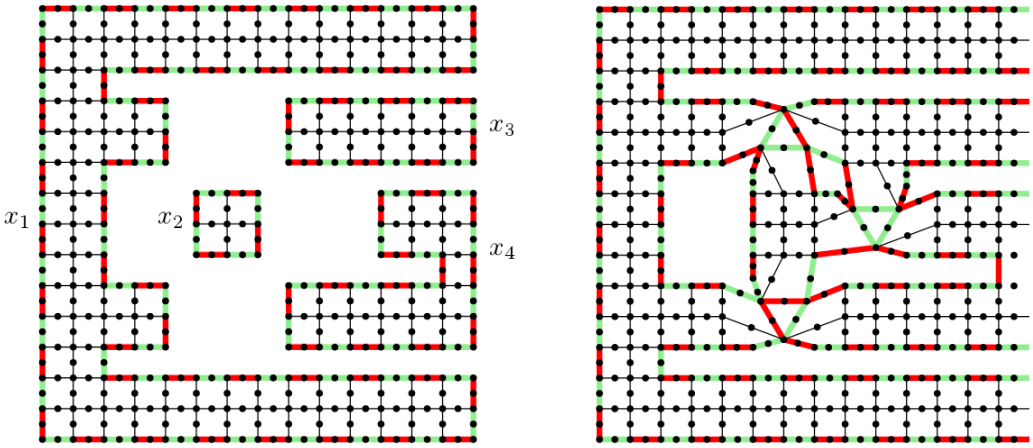


Figure 6: Instance of planar 3-SAT



(a) before contraction (b) after contraction

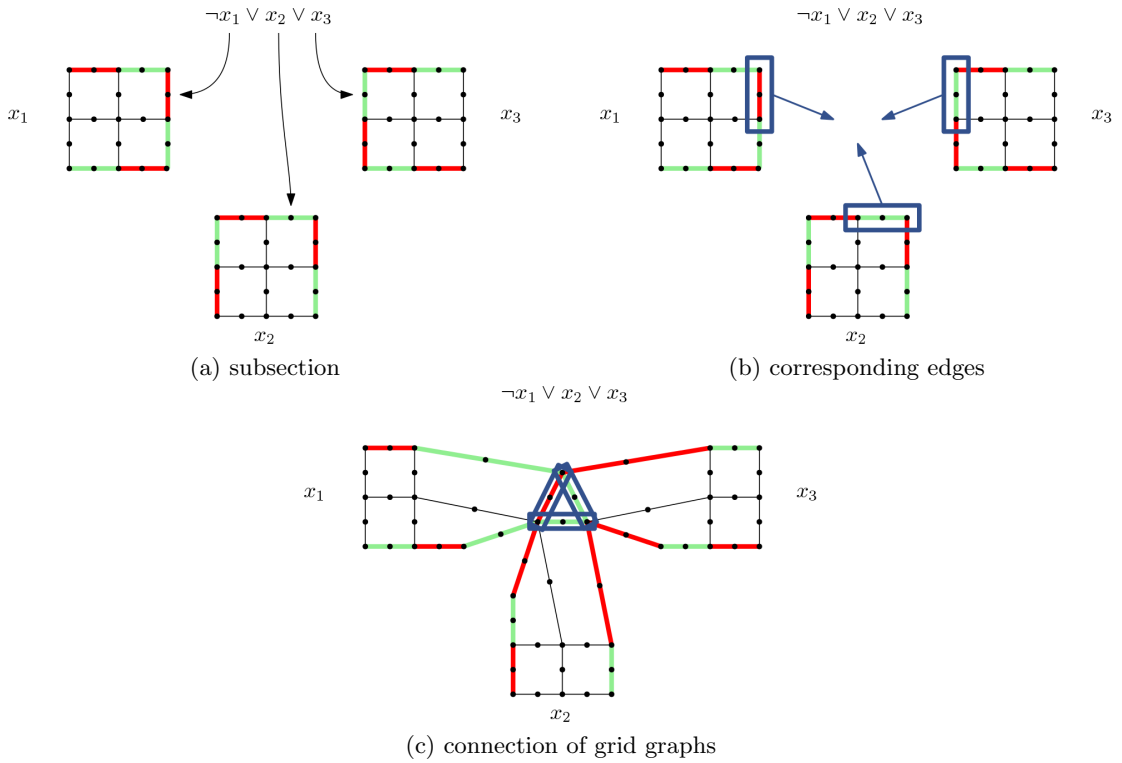
Figure 7: Example of a grid graph

5.1 NP-Hardness

An example of an instance of planar 3-SAT can be seen in figure 6 on page 7. The variables are represented by vertices and arranged on a spine in the middle of the graph. The clauses are also represented by vertices. Every clause is connected to all vertices that correspond to the variables that appear in the clause (normally or negated). The resulting graph has a planar embedding.

For the reduction we create a so called subdivided grid graph (a graph with an additional vertex on every edge). This additional vertex ensures that every edge in this graph is adjacent to one vertex with degree 2. We create such a grid graph for every variable of the original planar 3-SAT formula. These different grid graphs can be visualized in a rectangular fashion to further understanding, as seen in figure 7 on page 7.

It is helpful to imagine the grid graphs starting at the points on the spine of the 3-SAT visualization and then extruding along the edges until they reach a clause vertex where they end up in a situation like the one depicted in figure 8a. At this point it is important to mention that the outer boundary of the grid graphs is alternatingly colored red and green, however a change of color only occurs at a vertex of degree 3 or higher. The subdivision vertices lie therefore always on a uniformly colored edge. At the created meeting points of the grid graphs (each of the three meeting grid graphs corresponds to an occurring variable in the clause we try to model) we select an edge, according to the occurrence of the variable in the clause - positive or negative occurrence corresponds to a green or red edge,



(a) subsection

(b) corresponding edges

(c) connection of grid graphs

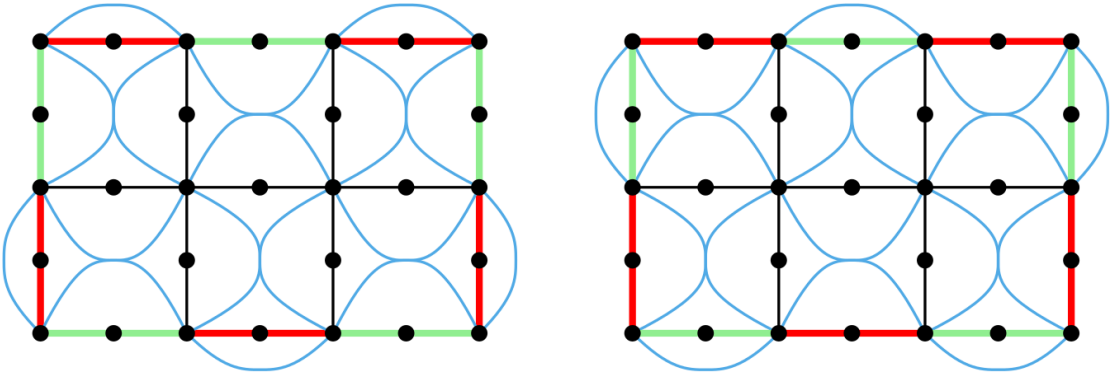
Figure 8: Grid graphs are connected corresponding to the occurrences of variables in a given clause

respectively (see figure 8b). Then we contract these edges like in figure 8c so that the three selected edges build a triangle after contraction. The result is the contracted grid graph of the *frame* F of the construction. all edges of F are adjacent to a degree-2 vertex and F has only one planar embedding (up to choice of the outer face).

The next step is the insertion of K_4's into the single cells of the Frame F. It should be paid attention to the fact that the cells are bounded by an 8-cycle which means that the vertices which are used to insert the K_4's are not the subdivision vertices. Note that a K_4 has only two different strict confluent drawings if all four vertices are drawn on the outside face. We can either construct a horizontal or a vertical junction in order to connect the diagonally separated vertices. After insertion in the frame, which then should be transformed into a strict confluent drawing, we can choose one of the two versions. However the neighboring cells are directly effected by this choice and have only one version left in order to maintain all adjacencies, as well as not creating multiple paths between vertices. this means that for a sufficiently large grid graph (at least 2 by 2 cells) the strict confluent drawing for the whole grid graph with inserted K_4's is determined by the choice for the first K_4 and there are only two different such drawings. Both can be seen in figure 9.

We will use these two different embeddings as an indicator if the corresponding variable for this grid graph is set to true or to false in a given variable assignment of the original 3-SAT problem. Note that it can be easily checked which version of the embedding was choosing since both embeddings have either all outside edges over a red (variable is false) or over a green edge (variable is true). These edges could also be drawn inside the grid cells, the edges over the opposite color (e.g. in the "true" assignment the edges over the red edge) can not be chosen since they are needed on the inside to correctly model adjacencies.

We will use the fact that only edges over edges with the same color can be drawn outside to our advantage in the following way. We insert a certain graph (which can be seen in figure 10 on page 9). This graph has the important property of not having a confluent drawing, which has all three of the edges, that connect the outer corners of the triangle, on the inside of the triangle. That means that at least one (possibly more) edges are drawn on the inside of the triangle in every possible strict confluent



(a) Version 1 (b) Version 2

Figure 9: Two possibilities of making a grid graph with inserted K_4's strict confluent

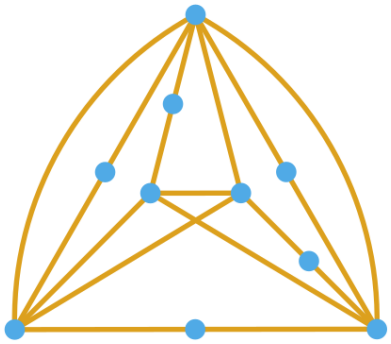


Figure 10: Clause graph

drawing. We will use this property to make a connection between variables which make this clause true and edges drawn on the inside of the triangle.

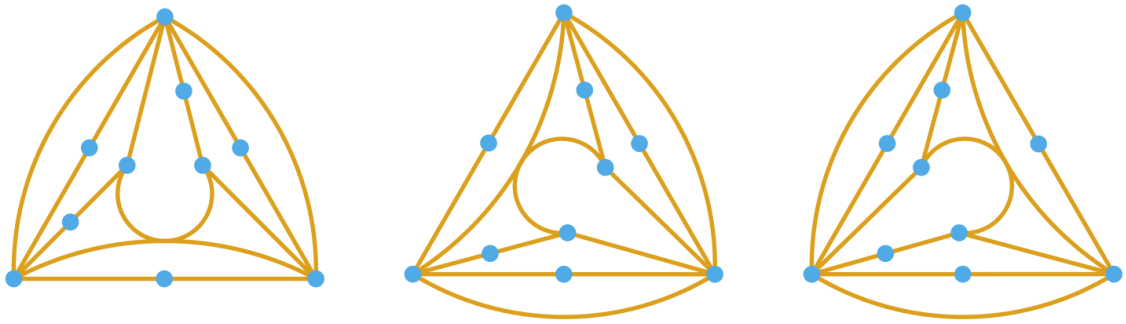
Lemma 5.1. *There is no strict confluent drawing of the clause graph in which all three triangle edges are drawn outside. Moreover, there is a strict confluent drawing of the clause graph with two of these edges outside, for every pair.*

Proof. Recall that by Lemma 3.1 the subdivided triangle must be embedded as a 6-cycle of direct arcs. To prove the first part of the lemma, assume that the triangle edges are all drawn outside this cycle. The remainder of the graph has no 4-cycles without subdivision vertices (that is, no $K_{2,2}$ with higher-degree vertices), so by Lemma 3.2 it can only have a strict confluent drawing if it is planar. However, it is a subdivided K_5, which is not planar. To prove the second part of the lemma, we refer to Figures 11a, 11b and 11c. □

Note that the two edges that are drawn outside in every drawing can easily be drawn on the inside without creating any crossings or violating any constraints, so this result in fact means, that one, two or all three of these edges are drawn inside but always at least one.

This concludes the construction of the graph out of an instance of planar 3-SAT. Whats left to show is that this created graph has a strict confluent drawing if and only if the planar 3-SAT formula from which it was created is satisfiable.

- If we have a satisfying variable assignment, the resulting graph has a strict confluent drawing. This can be shown by choosing the embedding option of the K_4's for every grid according to the truth value of the corresponding variable in the satisfying variable assignment If a variable x_i is set to *true*, we choose the embedding which enables the outer edges to be drawn (exclusively) over the green edges and we do the same for $false$ and the red edges. Since every clause was



(a) first (b) second (c) third

Figure 11: The three only possible confluent drawings of the clause graph depicted in figure 10 embeddings

modeled in such a way that the outer triangles correspond to the literals and their normal or negated occurrence in the clause, and we know that at least one variable in a positive literal is *true*, or one variable in a negative literal is $false$, we know that at least one edge can be drawn inside of the clause graph. According to Lemma 5.1 this means that the graph has a strict confluent drawing.

- If we have a strict confluent drawing, the planar 3-SAT has a satisfying variable assignment We use the same connection between edges being drawn inside of clause graphs and the corresponding truth assignment of that variable making the corresponding literal and therefore those clauses true. Since in a strict confluent drawing of the constructed graph every clause graph must have at least one edge drawn on the inside (again by Lemma 5.1) we can choose the variable assignment according to the the embedding of the K_4's. this variable assignment must fulfill the planar 3-SAT formula.

5.2 NP-Completeness

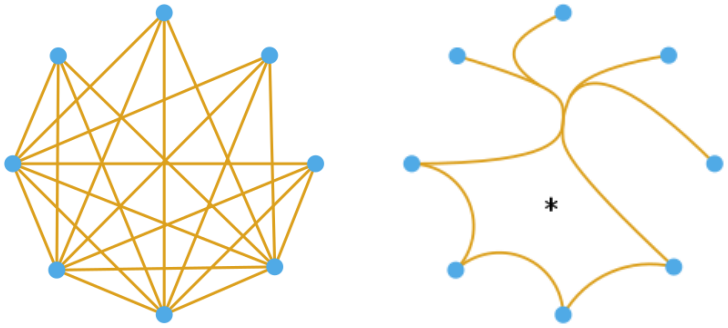
To show that testing strict confluence is in NP, recall that the combinatorial complexity of the drawing is linear in the number of vertices [7]. Thus the existence of a drawing can be verified by guessing its combinatorial structure and verifying that it is planar and a drawing of the correct graph.

Theorem 5.2. *Deciding whether a graph has a strict confluent drawing is NP-complete.*

6 Canonical diagrams

Eppstein et al. [7] also present an algorithm that recognizes if a graph has an outer planar strict confluent drawing. The algorithm makes use of an intermediate data structure called "canonical diagram". A canonical diagram serves as a representation of all possible outer planar strict confluent drawings. A given graph G might have multiple strict confluent drawings, as we have already seen earlier, but a canonical diagram has the property of being unique, if it exists. Furthermore a canonical diagram exists if and only if the corresponding graph has a strict confluent drawing [7]. Figure 12 on page 11 depicts a graph and its canonical representation.

Canonical diagrams consist (just as confluent drawings) of vertices, which are connected through a system of arcs and junctions. Adjacencies are again represented through the existence of a smooth curve from one vertex to another. However there is an additional possibility to model adjacency in a canonical diagram. A face f can be marked (in our case depicted with a star) which makes it possible to cross this face from sharp angle of an adjacent junction to another one, as if every junction on the boundary of f was connected by an arc.



(a) Graph (b) Canonical diagram

Figure 12: The right picture (b) is a representation of the node link graph on the left (a) as a canonical diagram

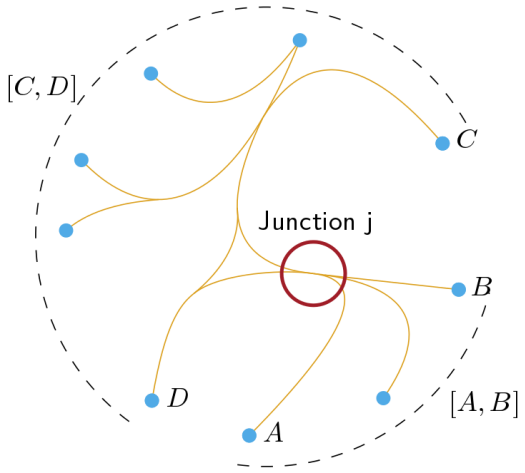


Figure 13: Funnel intervals of junction j

Also the following restrictions apply to canonical diagrams.

- Every arc is part of at least one smooth path from one vertex to another (which is allowed to cross marked faces from sharp angle to sharp angle).
- Any two smooth paths between the same two vertices must follow the same sequence of arcs and faces.
- Each marked face must have at least four angles, and all its angles must be sharp.
- Each arc must have either sharp angles or vertices at both of its ends.
- For each junction j with exactly two arcs in each direction, let f and f_0 be the two faces with sharp angles at j. Then it is not allowed for f and f_0 to both be either marked or a triangle (a face with three angles, all sharp).

And finally we need to define the funnel of a junction j of a canonical diagram D. A funnel is a 4-tuple of vertices a, b, c, and d where a and b are the vertex reached by a path that leaves j in one direction and continues as far clockwise/counterclockwise as possible, and c and d are the most clockwise/counterclockwise vertices reachable in the other direction. A funnel is depicted in figure 13 on page 11. Note that none of the paths from j to a, b, c, and d can intersect each other without contradicting the uniqueness of trails. We call the circular intervals of vertices $[a, b]$ and $[c, d]$ (in the counterclockwise direction) the funnel intervals of the respective funnel.

7 Algorithm

here we will present a high-level Overview of the before mentioned algorithm, that runs in $\mathcal{O}(n^2)$ time and $\mathcal{O}(n^2)$ space and recognizes if a graph has a outer-planar strict confluent drawing. The general procedure is as follows. The algorithm creates some additional data structures to speed up computations during the algorithm. Then it identifies all funnels in the input from which a list of all junctions of the canonical diagram is created. These junctions are then connected into a planar drawing which is a subset of the canonical drawing. Then it looks at the faces of the graph which was created in the step before and for each face identifies paths which would need to cross this face. This information is represented in a graph for each face. With these graphs the algorithm decides which connections can be drawn directly and which faces need to be marked. Finally it is checked if the result of the previous step aren't in conflict with the input and finally transforms the result into a strict confluent drawing. During this algorithm at several points, we control if the graph already exceeds bounds or violates in any way the restrictions we already identified and abort the construction algorithm, answering that there is no outer-planar strict confluent drawing.

For a detailed explanation of the used data structures and the complete algorithm we refer the reader to the original paper [7]. We will in the following look at some detailed point however, this overview will not be exhaustive of the whole algorithm.

- Conceptually, for each separated interval $[a, b]$, let c be the next neighbor of a that is counterclockwise of b, and let d be the next neighbor of b that is clockwise of a. If (i) c is a neighbor of b, (ii) d is a neighbor of a, (iii) a is the next neighbor of c that is counterclockwise of d, and (iv) b is the next neighbor of d that is clockwise of c, then (if a confluent diagram exists) a, b, c, d must form the funnel of a junction, and all funnels have this form. For a visual aid we refer again to figure 13 on page 11. With this method we can iterate over all circular intervals on the boundary of the graph in ascending order of cardinality to find every single possible funnel. Here we have the first check, which can result in an abortion of the algorithm, namely if the number of funnels exceeds a linear bound of Lemma 4.4.

- The found funnels can be transformed into junctions. Now for every vertex v we create a set of junctions which include v. These junctions are connected to v via a confluent tree. In this tree we can easily check if one junction is an ancestor of another one, or vice versa If neither is the case, the funnel intervals need to be disjoint and we can check which of the two vertices is clockwise of the other. This results in a planar embedding, which itself is part of the canonical diagram. Here we have the second check if we need to abort the algorithm, by computing the Euler characteristic of the result. We abort if the resulting embedding is not planar.

- Next we can create, for every face that was created in the step before, a graph H_f. This graph contains edges which are part of paths that need to cross the face f. We can find such edges by checking pairwise the junctions j and j' if there exists a path that travels through both junctions. We can do this by using a data structure which we didn't explicitly specify. If we find such a path, we draw an edge between j and j' in H_f. After doing this for all pairs of junctions on all faces, we draw edges of H_f which do not cross with any other edge. This creates new faces which then might contain crossing edges. Each such face is now marked.

- Finally we check if the created graph contains multiple edges between vertices (abort if thats the case), if the result correctly represents the adjacencies in the input (abort if thats not the case) and if the created diagram fulfills all criteria for canonical diagrams (abort if this is not the case).

- Now the algorithm is finished and has created the canonical diagram that corresponds to the input. We can now transform this canonical diagram into a strict confluent drawing by repeatedly pinching two non adjacent edges of a marked face together until the faces have only 3 adjacent junctions (they are now triangles).

The next theorem summarizes the properties of this algorithm.

Theorem 7.1. *For a given n-vertex graph G, and a given circular ordering of its vertices, it is possible to determine whether G has an outer-planar strict confluent drawing with the given vertex ordering, and if so to construct one, in time $\mathcal{O}(n^2)$.*

Without proof.

8 Conclusion

Eppstein et al. [7] have shown that, in confluent drawing, restricting attention to strict drawings allows us to characterize their combinatorial complexity. The recognition problem, whether a graph admits a strict confluent drawing has been shown to be NP-complete, but they have also shown that outer-planar strict confluent drawings with a fixed vertex ordering may be constructed in polynomial time.

There are multiple interesting open research questions. Perhaps the mos pressing problem [7] is to recognize the graphs that have an outer-planar strict confluent drawing without having a set order of the vertices and in what time they are recognizable. Which restrictions, other than outer planarity can we impose on strict confluent drawings such that they are still recognizable in polynomial time. Other question which have been mentioned are:

- Is every strict outer-planar confluent graph an alternation/circle polygon graph?
- Are graphs with rankwidth [16] 2, 3,... a subclass of outer-planar strict confluent?
- Has every permutation graph an outer-planar confluent drawing?

However, the overall most general question is, whether a given graph has a (not necessarily strict) confluent drawing, which is still unknown.

References

[1] Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. In *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2016. to appear.

[2] Weiwei Cui, Hong Zhou, Huamin Qu, Pak Chung Wong, and Xiaoming Li. Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1277–1284, 2008.

[3] Matthew Dickerson, David Eppstein, Michael T Goodrich, and Jeremy Yu Meng. Confluent drawings: Visualizing non-planar diagrams in a planar way. *J. Graph Algorithms Appl.*, 9(1):31–52, 2005.

[4] Tim Dwyer, Kim Marriott, and Michael Wybrow. Integrating edge routing into force-directed layout. In *International Symposium on Graph Drawing*, pages 8–19. Springer, 2006.

[5] David Eppstein, Michael T Goodrich, and Jeremy Yu Meng. Delta-confluent drawings. In *International Symposium on Graph Drawing*, pages 165–176. Springer, 2005.

[6] David Eppstein, Michael T Goodrich, and Jeremy Yu Meng. Confluent layered drawings. *Algorithmica*, 47(4):439–452, 2007.

[7] David Eppstein, Danny Holten, Maarten Löffler, Martin Nöllenburg, Bettina Speckmann, and Kevin Verbeek. Strict confluent drawing. In *Graph Drawing*, volume 8242, pages 352–363, 2013.

[8] Michael Hirsch, Henk Meijer, and David Rappaport. Biclique edge cover graphs and confluent drawings. In *International Symposium on Graph Drawing*, pages 405–416. Springer, 2006.

[9] Danny Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on visualization and computer graphics*, 12(5):741–748, 2006.

[10] Danny Holten and Jarke J Van Wijk. Force-directed edge bundling for graph visualization. In *Computer graphics forum*, volume 28, pages 983–990. Wiley Online Library, 2009.

[11] Peter Hui, Michael J Pelsmajer, Marcus Schaefer, and Daniel Stefankovic. Train tracks and confluent drawings. *Algorithmica*, 47(4):465–479, 2007.

[12] Christophe Hurter, Ozan Ersoy, and Alexandru Telea. Graph bundling by kernel density estimation. In *Computer Graphics Forum*, volume 31, pages 865–874. Wiley Online Library, 2012.

[13] TJ Jankun-Kelly, Tim Dwyer, Danny Holten, Christophe Hurter, Martin Nöllenburg, Chris Weaver, and Kai Xu. Scalability considerations for multivariate graph visualization. In *Multivariate Network Visualization*, pages 207–235. Springer, 2014.

[14] Stephen G Kobourov, Martin Nöllenburg, and Monique Teillaud. Drawing graphs and maps with curves (dagstuhl seminar 13151). In *Dagstuhl Reports*, volume 3. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[15] David Lichtenstein. Planar formulae and their uses. *SIAM journal on computing*, 11(2):329–343, 1982.

[16] Sang-il Oum and Paul Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006.

[17] Helen Purchase. Which aesthetic has the greatest effect on human understanding? In *International Symposium on Graph Drawing*, pages 248–261. Springer, 1997.

[18] Gianluca Quercini and Massimo Ancona. Confluent drawing algorithms using rectangular dualization. In *Graph Drawing*, volume 6502, pages 341–352. Springer, 2010.