Biclique Edge Cover Graphs and Confluent Drawings

Michael Hirsch, Henk Meijer, and David Rappaport

School of Computing, Queen's University
Kingston, Ontario, Canada

Abstract. Confluent drawing is a technique that allows some non-planar graphs to be visualized in a planar way. This approach merges edges together, drawing groups of them as single *tracks*, similar to train tracks. In the general case, producing confluent drawings automatically has proven quite difficult. We introduce the biclique edge cover graph that represents a graph G as an interconnected set of cliques and bicliques. We do this in such a way as to permit a straightforward transformation to a confluent drawing of G. Our result is a new sufficient condition for confluent planarity and an additional algorithmic approach for generating confluent drawings. We give some experimental results gauging the performance of existing confluent drawing heuristics.

1 Introduction

In 2003, Dickerson, Eppstein, Goodrich, and Meng introduced confluent drawing, and with it a heuristic able to generate confluent drawings for some graphs [1]. These drawings present a novel way of visualizing non-planar graphs in a planar way, however, producing a *planar confluent drawing* for an arbitrary graph has proven to be quite difficult. Devine speculates that merely deciding whether such a drawing exists is NP-hard for an arbitrary graph [1]. Hui, Schaefer, and Stefankovic also speculate that this problem is NP-complete [2]. In this paper we explore alternate methods of automatically generating confluent drawings. We experimentally evaluate Dickerson et al.'s confluent drawing heuristic, as well as our own heuristics based on the *biclique edge cover graph*.

Francis Newbery proposed a method of merging together edges called *edge concentration* in a 1989 paper [3]. Dickerson et al. first introduced confluent drawings in [3]; they have been subsequently studied in [2,4,1,5].

This paper is organized as follows. Section 2 provides a brief background, Sect. 3 defines the biclique edge cover graph, and Sect. 4 gives a method to transform such a graph into a confluent drawing. Finally, Sect. 5 covers confluent drawing algorithm implementations and their experimental performance.

2 Background

We define the relevant concepts in confluent drawing: A *curve* is a continuous map into the plane. A curve is *smooth* if it is continuously differentiable along

M. Kaufmann and D. Wagner (Eds.): GD 2006, LNCS 4372, pp. 405–416, 2007.
© Springer-Verlag Berlin Heidelberg 2007

its length (there are no sharp bends) [6,2]. A drawing D is a *confluent drawing* of an undirected graph G if:

- There is a one to one mapping between vertices of G and D.
- There exists a smooth curve between vertices u and w in D if and only if there exists an edge (u, w) in E.

Consistent with [4] and [1], we have omitted the planarity constraint found in Dickerson et al.'s definition [6]. We say that a confluent drawing is *planar* if no smooth curve(s) intersect at a single point (they may share overlapping portions). A confluent drawing is *non-planar* if any curve(s) intersect at a single point.

Lastly, we define a switch, and traffic circle, two basic confluent elements. A *switch* is the point where curves converge. A switch s is defined to have degree three [2,1], but we generalize it to have arbitrary degree. A *traffic circle* is a particular confluent representation of a clique such that all smooth curves merge with a central circular track. Figure 1 depicts a switch (left) and a traffic circle (right).

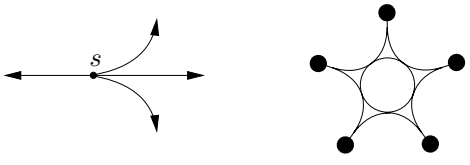


Fig. 1. A *switch* of degree four (left), and a confluent drawing of K_5 called a *traffic circle* (right)

3 Biclique Edge Cover Graph

Let $G = (V, E)$ be a graph. A clique c is a subset of V such that the subgraph induced by c is a complete graph. We say that edge e is in clique c if it is in the subgraph induced by the vertices in c.

A biclique (b_i, b_j) is an unordered pair of disjoint subsets b_i and b_j of V, such that for all $u \in b_i$ and $w \in b_j$, $(u, w) \in E$. We call each subset b_i and b_j a *b-part*. We say that edge e is in biclique (b_i, b_j) if it is incident to a vertex in each b-part.

Let C be a set of cliques, and let B be a set of bicliques such that each edge of G is in a clique of C or in a biclique of B. We say that such sets B and C together *edge cover* G. Given a set of bicliques B, B_p is the set of b-parts such that for each $(b_i, b_j) \in B$, $b_i, b_j \in B_p$.

Let G be a graph. Let B be a set of bicliques and let $C = \{c_0, c_1, \ldots, c_{m-1}\}$ be a set of cliques that together edge cover G. Let b-parts $b_0, b_1, \ldots, b_{l-1}$ denote the elements of B_p. Let $\pi_0, \pi_1, \ldots, \pi_{2^{l+m}-1}$ denote the elements in the power set of $B_p \cup C$. We define the biclique edge cover graph $G_b = (V_b, E_b)$ as follows:

- Vertex $v_b = \pi_i$ is in V_b if and only if there exists a vertex $v \in V$ such that v is in every b-part and clique in π_i and no others.
- Edge $e_b = (u_b, w_b)$ is in E_b if and only if $u_b \neq w_b$, and $u_b \cap w_b \cap C \neq \emptyset$, or there exists a $b_i \in u_b$ and $b_j \in w_b$ such that $(b_i, b_j) \in B$.

We say that v and v_b are associated if a vertex $v \in V$ is in every b-part and clique in $v_b = \pi_j \in V_b$ and no others. Hereafter, u, v, w and $v_0, v_1, \ldots, v_{n-1}$ denote the elements of V. Similarly, u_b, v_b, w_b and $v_{b_0}, v_{b_1}, \ldots, v_{b_{n-1}}$ denote the elements of V_b.

3.1 Example

The following example illustrates the derivation of a biclique edge cover graph G_b from graph G (Fig. 2). Given a graph G, first determine a set of bicliques B, and cliques C. Note that any sets will suffice, provided that B and C together edge cover G. We choose $B = \{(\{v_0, v_1\}, \{v_2, v_3, v_4\})\}$ and $C = \{c_0\} = \{v_2, v_3, v_4, v_5, v_6\}$. Thus $B_p = \{b_0, b_1\} = \{\{v_0, v_1\}, \{v_2, v_3, v_4\}\}$.

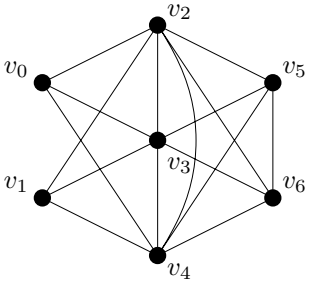


Fig. 2. Graph G

We construct the vertex set V_b: The vertex $v_0 \in b_0$, $v_0 \notin b_1 \cap c_0$. By our definition, v_0 establishes $\{b_0\} \in V_b$, and v_0 and $\{b_0\}$ are associated. Vertex v_1 also establishes $\{b_0\} \in V_b$. Vertices v_2, v_3, v_4 each establish $\{b_1, c_0\} \in V_b$. Vertices v_5 and v_6 establish $\{c_0\} \in V_b$. Thus $V_b = \{v_{b_0}, v_{b_1}, v_{b_2}\} = \{\{b_0\}, \{b_1, c_0\}, \{c_0\}\}$. The edge set $E_b = \{(v_{b_1}, v_{b_2}), (v_{b_0}, v_{b_1})\}$. Figure 3 depicts biclique edge cover graph G_b. We use a solid vertex to depict any $v_b \cap C \neq \emptyset$, and an unfilled vertex to depict any $v_b \cap C = \emptyset$.

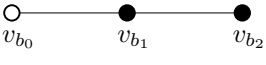


Fig. 3. Derived biclique edge cover graph G_b

Lemma 1. *Let G be a graph. Let B be a set of bicliques and let C be a set of cliques that together edge cover G. Let G_b be the resulting biclique edge cover graph. The following properties hold:*

1. *The magnitude $|V_b| \leq |V|$.*
2. *Vertices $v_0, v_1, \ldots, v_{r-1} \in V$ associated with a vertex $v_b \in V_b$ define an independent set in G where $v_b \cap C = \emptyset$; otherwise, they define a clique in G.*

Proof. Property (1) follows from our definition of a biclique edge cover graph: Each vertex $v \in V$ may only be associated with a single vertex $v_b \in V_b$. Each vertex $v_b \in V_b$ is associated with at least one vertex $v \in V$. It follows that $|V_b| \leq |V|$.

We prove Property (2) by contradiction. Let u and w be adjacent vertices in V, such that u and w are associated with $v_b \in V_b$ and $v_b \cap C = \emptyset$. By definition, edge $e = (u, w)$ is in a biclique in B or a clique in C. If e is in a clique $c \in C$, then c must be an element of v_b. This is contrary to $v_b \cap C = \emptyset$. If e is in a biclique $(b_i, b_j) \in B$, it follows that u must be an element of b_i and w an element of b_j. Recall that b-parts b_i and b_j are disjoint. Vertices u and w cannot therefore both be associated with v_b. Where $v_b \cap C \neq \emptyset$, Property (2) follows directly from the definition. □

4 Generating Confluent Drawings

In this section we show how to construct a drawing D of G from a drawing of its biclique edge cover graph G_b. Let G be a graph. Let B be a set of bicliques and let C be a set of cliques that together edge cover G. Let G_b be the resulting biclique edge cover graph. Let D_b be a drawing of G_b. Note that drawing D_b could be a traditional drawing or a confluent drawing of G_b. Replace each vertex $v_{b_i} \in D_b$ by vertices of G as follows:

We will compose a confluent structure. Begin with a single circular track. Join each vertex in V associated with v_{b_i} to the circular track:

- If $v_{b_i} \cap C \neq \emptyset$: Join each vertex in V associated with v_{b_i} to the circular track by means of two smooth curves such that one curve may be followed onto, and around the adjoined circular track in the clockwise direction, and the other in the counterclockwise direction. We call this construction a *traffic circle*.
- If $v_{b_i} \cap C = \emptyset$: Join each vertex in V associated with v_{b_i} to the circular track by means of a smooth curve such that the curve may be followed onto, and around the adjoined circular track in the counterclockwise direction. We call this construction a *counterclockwise traffic circle*.

Remove vertex v_{b_i} from drawing D_b, and put the composed confluent structure in its place. Merge all (confluent) edges previously incident to v_{b_i} with the circular track such that each edge may be followed onto, and around the adjoined

circular track in the clockwise direction. Figure 4 illustrates our replacement method for a vertex $v_{b_i} \in D_b$ with two incident edges, and three associated vertices in V.

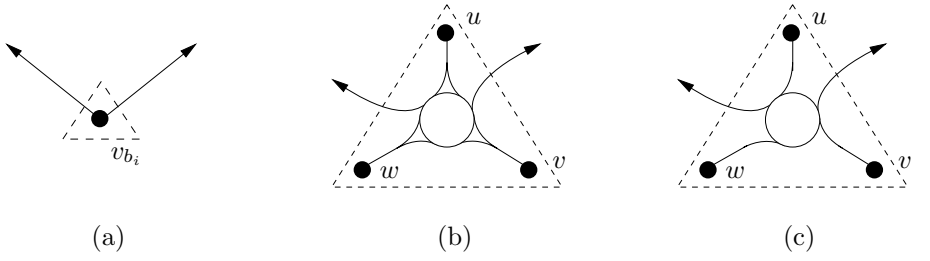


Fig. 4. (a) Vertex v_{b_i} in drawing D_b (b) Replacement of v_{b_i} in drawing D where $v_{b_i} \cap C \neq \emptyset$ (c) Replacement of v_{b_i} in drawing D where $v_{b_i} \cap C = \emptyset$

4.1 Example

We continue with the example of Sect. 3.1. We generate a confluent drawing of G from a drawing of G_b (Fig. 3). Each vertex of G_b is replaced by all associated vertices of G. The result is drawing D, Fig. 5.

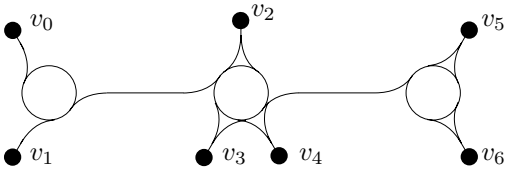


Fig. 5. Confluent drawing D generated from a drawing of G_b

Lemma 2. *Let D be a drawing generated from a drawing of G_b by the method of this section. Drawing D is a confluent drawing of G.*

Proof. We first show that a one to one mapping exists between vertices of G and D. We then show that there exists a smooth curve between vertices u and w in D if and only if there exists an edge (u, w) in E. We present this argument in two cases.

Our method replaces each vertex $v_b \in V_b$ by all associated vertices in V. Because each vertex in V is associated with a single vertex in V_b, the vertex set of D is precisely that of G.

Case I. We will show that there exists a smooth curve between vertices u and w in D if there exists an edge (u, w) in E. Vertices u and w are either associated with the same vertex, or two different vertices in V_b. If they are associated with

the same vertex $v_{b_i} \in V_b$ and $v_{b_i} \cap C \neq \emptyset$, then our method sees that u and w are in the same constructed traffic circle. A smooth curve therefore exists between u and w in D. Otherwise (if $v_{b_i} \cap C = \emptyset$) Lemma 1 precludes (u, w) from being an edge in E.

We now show that a smooth curve exists between u and w if they are associated with different vertices in V_b: u with u_b and w with w_b. By definition, edge (u, w) is in a biclique in B or a clique in C. If (u, w) is in a clique $c \in C$, then clique c is an element of both u_b and w_b. Vertices u_b, w_b are therefore adjacent. If (u, w) is in a biclique $(b_i, b_j) \in B$, then b_i is an element of u_b and b_j an element of w_b. Vertices u_b, w_b are again adjacent. Our method ensures that the (confluent) edge between u_b and w_b is merged with the circular track that replaces u_b and the circular track that replaces w_b. This merged edge completes a smooth curve between u and w in D.

Case II. We will show that there exists an edge (u, w) in E if there exists a smooth curve between vertices u and w in D. Our method generates smooth curves in D in two ways. First, constructed traffic circles consist of smooth curves between vertices of V. Two vertices u and $w \in V$ are in the same traffic circle only if they are associated with the same vertex in V_b. It follows from Lemma 1 that edge $(u, w) \in E$.

Additionally, smooth curves connect traffic circles/circular tracks that have replaced adjacent vertices in V_b. Because (confluent) edges are always merged with these confluent structures in the same direction, a smooth curve never connects two structures that have replaced non-adjacent vertices in V_b. Smooth curves in D therefore connect vertices u and w that are associated with adjacent vertices in V_b. If vertices $u_b, w_b \in V_b$ are adjacent, then there exists a clique $c \in u_b \cap w_b$ or there exist b-parts $b_i \in u_b$ and $b_j \in w_b$ such that $(b_i, b_j) \in B$. If $c \in u_b \cap w_b$, then any vertex associated with either u_b or w_b is in c. Otherwise, if $b_i \in u_b$ and $b_j \in w_b$, then u is an element of b_i and w an element of b_j. In either case, it follows that edge $(u, w) \in E$. □

4.2 Planarity

Lemma 3. *Let D be a drawing generated from a drawing of G_b by the method of this section. If the drawing of G_b is confluent planar then D is confluent planar.*

Proof. Our method replaces vertices in the drawing of G_b by confluent planar structures to produce D. No edge crossings exist in the drawing of G_b, and none are introduced by our replacement scheme. Drawing D is therefore a confluent planar drawing of G. □

Corollary 1. *If the drawing of G_b is planar then D is confluent planar.*

Proof. A planar graph satisfies the definition of confluent planarity. □

4.3 Prickly Clique

A *prickly clique* consists of a clique and one additional vertex adjacent to each vertex in the clique. More formally, a prickly clique is a graph G on $2n$ vertices

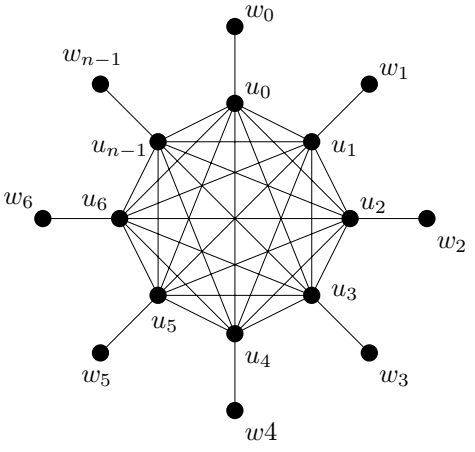


Fig. 6. Prickly clique G for $n = 8$

with $n \geq 2$ such that $V = \{u_0, u_1, \ldots u_{n-1}, w_0, w_1, \ldots, w_{n-1}\}$, and $E = E_0 \cup E_1$ where $(u_i, w_i) \in E_0$ for $0 \leq i < n$ and $(u_i, u_j) \in E_1$ for $0 \leq i < j < n$. The prickly clique for $n \geq 5$ is an example of a confluent planar graph that does not have a resulting planar biclique edge cover graph.

Lemma 4. *Let G be a prickly clique. Let B be a set of bicliques and let C be a set of cliques that together edge cover G. Let G_b be the resulting biclique edge cover graph. Graph G is isomorphic to G_b.*

Proof. We construct the vertex set V_b. Each edge $(u_i, w_i) \in E_0$ is either a clique in C or a biclique in B, while each edge $(u_i, u_j) \in E_1$ is in at least one biclique in b or clique in C. Thus, each vertex $w_i \in G$ establishes a vertex $w_{b_i} \in V_b$, while each vertex $u_i \in G$ establishes a vertex $u_{b_i} \in V_b$. This defines a bijection $u_i \rightarrow u_{b_i}$, $w_i \rightarrow w_{b_i}$, from V to V_b.

We construct the edge set E_b. Edge $(u_{b_i}, w_{b_i}) \in E_b$ for $0 \leq i < n$ (if $\{u_i, w_i\} \in C$ then $u_{b_i} \cap w_{b_i} \cap \{u_i, w_i\} \neq \emptyset$; otherwise if $\{\{u_i\}, \{w_i\}\} \in B$ then $\{u_i\} \in u_{b_i}$ and $\{w_i\} \in w_{b_i}$). Moreover, edge $(u_{b_i}, u_{b_j}) \in E_b$ for $0 \leq i < j < n$ (if (u_i, u_j) is in a clique $c \in C$ then $u_{b_i} \cap u_{b_j} \cap c \neq \emptyset$; otherwise if (u_i, u_j) is in a biclique $b = (b_i, b_j) \in B$ then $b_i \in u_{b_i}$ and $b_j \in u_{b_j}$). □

5 Implementation and Results

In this section we examine two algorithmic approaches for generating confluent drawings. We will examine the experimental performance of each implemented algorithm, and conclude with some sample outputs.

5.1 ConfluentDickerson(G)

Algorithm 1 is based on the only previously published confluent drawing algorithm for undirected graphs. Presented by Matthew Dickerson at the 11th International Symposium on Graph Drawing, the heuristic iteratively identifies and replaces large cliques and bicliques (complete and bipartite subgraphs) with equivalent confluent structures [6].

Our implementation uses an $O(nm\mu)$ solution (where μ is the number of maximal independent sets of a graph) to the maximal independent sets problem by Tsukiyama et al. [7] to enumerate all maximal cliques and bicliques. This solution yields all maximal cliques when applied to a graph's complement, and all maximal bicliques when applied to its *double cover* [8]. Note that Dickerson et al. [6] specified an algorithm by Chiba and Nishizeki [9] for identifying cliques and a second algorithm by Eppstein [10] for identifying bicliques.

Algorithm 1: ConfluentDickerson(G)

Input: A connected graph $G = (V, E)$
Output: A confluent drawing of G

```
done ← false;
while !done and G is non-planar do
    C ← all maximal cliques of G;
    foreach clique c ∈ C in order of decreasing size do
        if there exists an edge in E between each pair of vertices in c then
            Remove all edges from E between pairs of vertices in c;
            Add a vertex u to V; denote it as a traffic circle; Add an edge to
            E between u and each vertex in c;
            done ← false;
    B ← all maximal bicliques of G;
    foreach biclique (b_i, b_j) ∈ B in order of decreasing size do
        if there exists an edge in E between each pair of vertices (v, w),
        where v ∈ b_i, w ∈ b_j then
            Remove each edge (v, w) from E where v ∈ b_i, w ∈ b_j;
            Add vertices v and w to V; denote each as a switch;
            Add an edge (v, w) to E;
            Add an edge to E between v and each vertex in b_i;
            Add an edge to E between w and each vertex in b_j;
            done ← false;
Draw G;
```

5.2 ConfluentHirsch(G)

Algorithm 2 is an implementation of the algorithm presented in Secs. 3 and 4. The algorithm first randomly computes a set of cliques and bicliques that

together edge cover G. It then determines the vertex set of the biclique edge cover graph. Each of these vertices is inserted into G, joined to all associated vertices, and denoted as a *traffic circle* or *counterclockwise traffic circle*.

RecursiveHirsch(G, i). Beginning with G, this variation of $ConfluentHirsch$ (G) iteratively computes i successive biclique edge cover graphs. A confluent drawing of G is recursively constructed using the algorithm in Sec 4. See [11] for details.

DiscardHirsch(G). This second variation discards cliques $|c| \leq 3$ from C and bicliques $|b_i|, |b_j| \leq 1$ from B. Intuitively, including these degenerate cases can only hamper the performance of our algorithm. Any vertices that are no longer in a clique or biclique after the discard are effectively ignored by the algorithm.

Algorithm 2: ConfluentHirsch(G)

Input: A connected graph $G = (V, E)$
Output: A confluent drawing of G

Let V be an array of vertices of G and let V_b be an array of collections;
Let C be a collection of cliques, and let B be a collection of bicliques;
Let B_p be a collection *b-parts* such that for all $(b_i, b_j) \in B$, $b_i, b_j \in B_p$;
*A collection is a single object that contains multiple elements.

foreach *edge* $e \in E$ **do**
  if *edge* e *is not yet covered by a clique* $\in C$ *or a biclique* $\in B$ **then**
    Randomly expand e into a maximal clique or biclique and accordingly add it to C or B;
Remove all edges from E;
foreach *set* $s \in C \cup B_p$ **do**
  foreach *vertex* $v \in s$ **do**
    Add set s to $V_b[V.indexOf(v)]$;
foreach *unique collection* $u_b \in V_b$ **do**
  if $u_b \cap C \neq \emptyset$ **then**
    Add a vertex u_b to V; denote it as a *traffic circle*;
  else if $u_b \cap C = \emptyset$ **then**
    Add a vertex u_b to V; denote it as *counterclockwise traffic circle*;
  Add an edge to E between vertex u_b and each vertex v where $V_b[V.indexOf(v)] = u_b$;
Add an edge to E between any two vertices u and w where $u \cap w \cap C \neq \emptyset$;
Add an edge to E between any two vertices u and w where $u \in b_i$ and $w \in b_j$ such that $(b_i, b_j) \in B$;
Draw G;

5.3 Experimental Results

We have applied our drawing algorithm implementations to two sets of graphs in order to measure their performance[1]. Table 1 summarizes results for the *Rome graphs* [12], and Table 2 for the *ATT graphs* (http://www.graphdrawing.org).

Because ConfluentHirsch(G) and DiscardHirsch(G) are non-deterministic, they were allowed multiple attempts per input to produce a confluent planar output. For a given case, a single confluent planar output was recorded if at least one attempt produced a confluent planar output. Note that RecursiveHirsch(G, i) is also non-deterministic, however, multiple recursive iterations ensure that its output is not determined by one random set of cliques and bicliques.

Table 1. Performance of confluent drawing algorithms on the Rome set

Algorithm applied	Non-planar inputs	Attempts per input	Confluent planar outputs	Confluent planar outputs not found by ConfluentDickerson(G)
ConfluentDickerson(G)	8253	1	210	-
ConfluentHirsch(G)	8253	10	9	1
ConfluentHirsch(G)	8253	100	10	1
RecursiveHirsch(G,10)	8253	1	10	1
RecursiveHirsch(G,100)	8253	1	10	1
DiscardHirsch(G)	8253	10	115	22

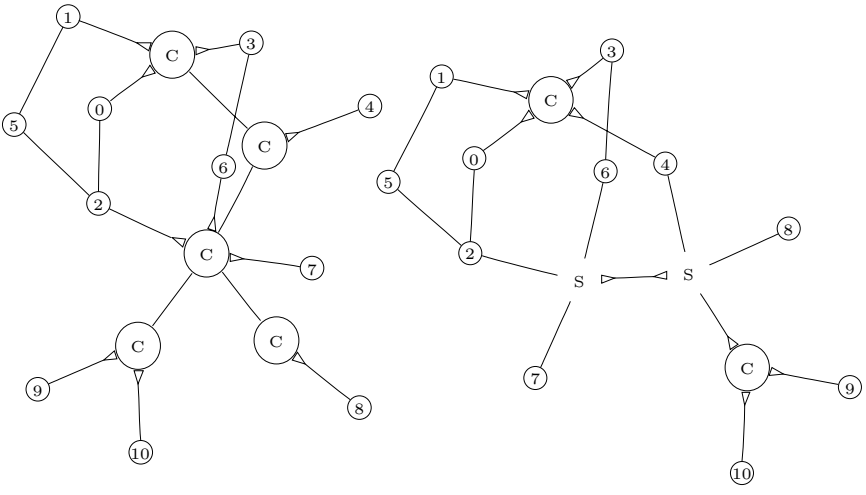
Table 2. Performance of confluent drawing algorithms on the ATT set

Algorithm applied	Non-planar inputs	Attempts per input	Confluent planar outputs	Confluent planar outputs not found by ConfluentDickerson(G)
ConfluentDickerson(G)	423	1	166	-
ConfluentHirsch(G)	423	10	48	0
ConfluentHirsch(G)	423	100	53	0
RecursiveHirsch(G,10)	423	1	57	1
RecursiveHirsch(G,100)	423	1	61	2
DiscardHirsch(G)	423	10	129	5

Figures 7 and 8 were output by our implementation.[2] Switches are denoted S, with an arrowhead marking the incident edge along which the other incident edges converge. Traffic circles are denoted C:

[1] Planarity was determined using the Lempel-Even-Cederbaum planarity test implementation included as part of GTL, the *Graph Template Library* (http://www. infosun.fmi.uni-passau.de/GTL).

[2] Our implementation uses the *Graphviz* [13] package to produce layouts and drawings, as well as the *Grappa* [14] package for working with graphs in Java.



(a) DiscardHirsch(G) (b) ConfluentDickerson(G)

Fig. 7. Confluent drawings of an example given by Dickerson et al. in [6]. Spring embedder layout computed using *Graphviz* [13].

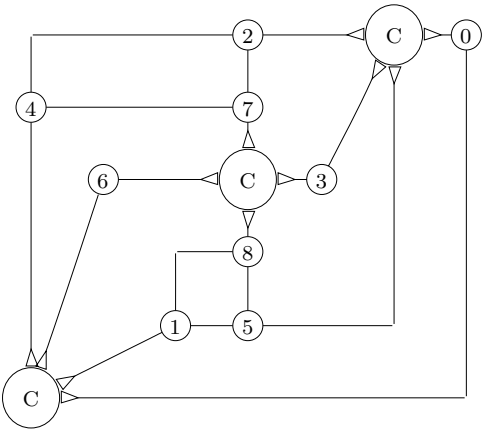


Fig. 8. The smallest known confluent non-planar graph is the Peterson graph with one vertex and adjacent edges removed [12]. Above, a confluent planar drawing of its complement generated by ConfluentDickerson(G). Layout computed using the dominance-polyline method for general undirected planar graphs in [15].

6 Conclusion

The performance of the algorithms varied, with ConfluentDickerson(G) producing the greatest number of confluent planar drawings for both sets of graphs.

Each variation of ConfluentHirsch(G) was however able to produce confluent planar results for some inputs where ConfluentDickerson(G) was not. Our results seem to confirm that confluent drawings offer a valid means for drawing non-planar graphs in a planar way for some inputs. Confluent drawings can however be more difficult to read than traditional drawings. This holds true even for cases where a confluent drawing is planar and the original graph is not.

References

1. Devine, J.: Confluent graphs. Master's thesis, Queen's University (2005)
2. Hui, P., Schaefer, M., Štefankovič, D.: Train tracks and confluent drawings. In Pach, J., ed.: Proc. 12th Int. Symp. Graph Drawing (GD 2004). Number 3383 in Lecture Notes in Computer Science, Springer-Verlag (2004) 318–328
3. Newbery, F.J.: Edge concentration: a method for clustering directed graphs. In: Proc. 2nd Int. Works. on Soft. configuration management, ACM Press (1989) 76–85
4. Eppstein, D., Goodrich, M.T., Meng, J.Y.: Confluent layered drawings. In Pach, J., ed.: Proc. 12th Int. Symp. Graph Drawing (GD 2004). Number 3383 in Lecture Notes in Computer Science, Springer-Verlag (2004) 184–194
5. Eppstein, D., Goodrich, M.T., Meng, J.Y.: Delta-confluent drawings. In Healy, P., Nikolov, N.S., eds.: Proc. 13th Int. Symp. Graph Drawing (GD 2005). Number 3843 in Lecture Notes in Computer Science, Springer-Verlag (2006) 165–176
6. Dickerson, M.T., Eppstein, D., Goodrich, M.T., Meng, J.Y.: 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)
7. Tsukiyama, S., Ide, M., Ariyoshi, H., Shirakawa, I.: A new algorithm for generating all the maximal independent sets. In: SIAM Journal on Computing. Volume 6. (1977) 505–517
8. Alexe, G., Alexe, S., Crama, Y., Foldes, S., Hammer, P.L., Simeone, B.: Consensus algorithms for the generation of all maximal bicliques. Discrete Applied Mathematics **145**(1) (2004) 11–21
9. Chiba, N., Nishizeki, T.: Arboricity and subgraph listing algorithms. SIAM J. Comput. **14**(1) (1985) 210–223
10. Eppstein, D.: Arboricity and bipartite subgraph listing algorithms. Information Processing Letters **51**(4) (1994) 207–211
11. Hirsch, M.: Generating confluent drawings: Theory and practice. Master's thesis, Queen's University (2006)
12. Di Battista, G., Garg, A., Liotta, G.: An experimental comparison of three graph drawing algorithms (extended abstract). In: Proc. 11th Annual Symp. Computational Geometry (SCG '95), New York, NY, USA, ACM Press (1995) 306–315
13. Gansner, E.R., North, S.C.: An open graph visualization system and its applications to software engineering. Softw. Pract. Exper. **30**(11) (2000) 1203–1233
14. Barghouti, N.S., Mocenigo, J., Lee, W.: Grappa: A graph package in java. In Di Battista, G., ed.: Proc. 5th Int. Symp. Graph Drawing (GD '97). Volume 1353 of Lecture Notes in Computer Science., Springer (1997) 336–343
15. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, NJ, USA (1998)