Drawing cycles in networks

Christoph Buchheim, Michael Jünger, Merijam Percan, Michael Schulz, and Christina Thelen

Universität zu Köln, Institut für Informatik, Pohligstraße 1, 50969 Köln, Germany,
{buchheim,mjuenger,percan,schulz,thelen}@informatik.uni-koeln.de,
partially supported by Marie-Curie Research Training Network (ADONET) and by the German Science Foundation (JU204/10-1).

Abstract. In this paper we show how a graph that contains a specified cycle can be drawn in the plane such that the cycle is drawn circularly while the rest of the graph is layouted orthogonally.
We also show how to extend this algorithm to deal with a set of disjoint cycles at once.

1 Introduction

Biochemical networks possess a complex structure and a big information content. Visualization of the networks can help the scientists to understand the structure in a short time while a textual representation does not offer this quick access.

To create such drawings by hand is very time-consuming. Michal [4] needed several years to create his poster *Biochemical Pathways*. Hence, it is a good idea to use an automatic graph drawing tool to create the actual layout. This is encouraged by the development of the graphs over time, as new data is added every day, and by the possibility to manipulate the graphs by hand. On the other hand, biologists like to recognize special substructures easily by watching out for special sub-drawings. For example, the citrate cycle can be easily recognized in an orthogonal drawing when it is drawn circularly. Thus, existent graph drawing algorithms that do not yet fulfill the special wishes of biologists need to be adapted.

2 Drawing cycles in networks

In this section we present an algorithm that draws a graph in the plane such that a given cycle of the graph is drawn circularly and the rest of the graph is drawn using an orthogonal layout. The input of the algorithm consists of a graph $G = (V, E)$, together with a set $\mathcal{C}$ of disjoint cycles $C \subseteq G$. The algorithm is restricted to a special class of graphs as input, so-called cycle/face-planar graphs. We give a characterization of this class in Section 2.1. In Section 2.3 we discuss how the input of the algorithm can be extended to general graphs by introducing a suitable preprocessing step.

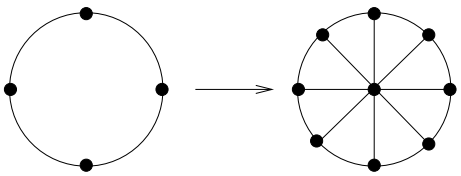
2.1 Cycle/face-planar graphs

Given an undirected planar graph $G = (V, E)$ and a set $\mathcal{C}$ of disjoint cycles of G, we call the pair $(G, \mathcal{C})$ *cycle/face-planar*, if there is a planar embedding Π of G such that each cycle $C \in \mathcal{C}$ borders a face in Π. Π is called cycle/face-planar embedding. We show how a pair $(G, \mathcal{C})$ can be tested to be cycle/face-planar in linear time.

As planarity can be tested in linear time, we can assume G to be planar. Obviously, each cycle $C \in \mathcal{C}$ may appear in exactly one biconnected component, called block, of G since C is a biconnected subgraph of G. Consequently, we may reduce the problem to decide cycle/face-planarity to instances where G is a biconnected planar graph. Hence, G is cycle/face-planar if and only if every block of G is cycle/face-planar.

Having a biconnected planar graph G at hand the question that arises is whether there is a transformation of G into an arbitrary graph H such that G is cycle/face-planar if and only if H is planar.

Theorem 1. *Given a biconnected planar graph $G = (V_G, E_G)$ with a set $\mathcal{C}$ of cycles, let $H = (V_H, E_H)$ be a biconnected graph that is constructed iteratively from G by replacing each cycle of $\mathcal{C}$ with an extended wheel as shown in the following figure:*



Then G is cycle/face-planar if and only if H is planar.

Proof. Assume we are given a cycle/face-planar pair $(G, \mathcal{C})$ in which G is biconnected and planar. Let Π be a cycle/face-planar embedding of G. By definition, every cycle $C \in \mathcal{C}$ borders a face in Π. Splitting the edges of C in Π (and thus enlarging the cycle by creating twice as many vertices and edges) does not have any effect on the cycle/face-planarity since it still borders the same face again. Hence, we get a cycle/face-embedding Π' and a modified cycle C'. The addition of a center vertex v inside of C' in Π' and the addition of edges to each vertex in C' (and thus transforming the cycle to a wheel) destroys the cycle/face-planarity but it obviously keeps planarity. Doing this transformation in Π for every cycle of C we get a new graph H of G that is planar.

Assume now we are given a planar graph H of G that is constructed in the described way. By deleting the spokes of the extended wheels and shrinking every second edge of the resulting cycles in a planar embedding of H we get a cycle/face-planar embedding of G. Indeed, the spokes connecting the center with the new vertices on the rim ensure that no block of G ends up in the interior of the cycle. Consequently, G is cycle/face-planar. □

We call H the *cycle/face-graph* of G. Since H can be constructed in linear time we can test cycle/face-planarity in linear time. Obviously, a cycle/face-planar embedding is a planar embedding.

Given a cycle/face-planar pair $(G, \mathcal{C})$ we investigate the complexity of getting a cycle/face-planar embedding. For a biconnected graph we transform G into its cycle/face-graph H, get any planar embedding of H and establish a cycle/face-planar embedding of G by retransforming H back to G.

If G is connected but not biconnected, we need to take care of cut vertices that belong to any cycle. In this case we embed the blocks outside of the cycles by simply adjusting the adjacency lists of the cut vertices (see Figure 1).

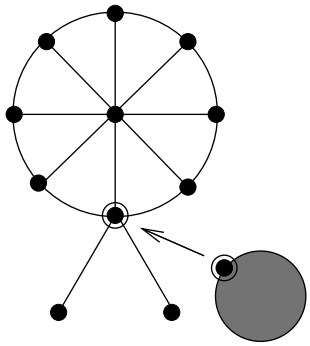


Fig. 1. Construction of a cycle/face-planar embedding in the connected but not biconnected case; the circled vertex is a cut vertex that is contained, both, in a cycle of $\mathcal{C}$ and in a different block (visualized grey).

The case of a disconnected graph can be reduced to the connected case as every connected component can be dealt with independently.

2.2 A new drawing algorithm

Tamassia et al. [5] developed an algorithm for drawing a 4-planar graph, i.e., a graph with vertex degrees at most 4 with a given planar embedding. All edges are drawn orthogonally and the number of edge bends is minimized. Each vertex is represented by a rectangle such that each side of the rectangle has at most one outgoing edge.

Klau [3] extended this algorithm to create quasi-orthogonal drawings for general graphs. As the number of edges incident to a vertex is not limited anymore, he surrounded vertices with degrees greater than 4 by a rectangular box of sufficient size such that more than one edge can be connected to a single side of a box. The box of a vertex v with degree d is represented by a cycle of length d such that each cycle vertex is connected to one of the incoming edges (and has hence degree 3). This modified graph is drawn with the original algorithm for

4-planar graphs with the restriction that the constructed cycles are drawn in a rectangular shape. In a post-processing step the box is substituted by a single vertex in the center representing the original vertex and connecting edges from the center vertex to the cycle vertices that are transformed into edge bends.

Changes We assume our input to be a cycle/face-planar graph G and a cycle $C \subseteq G$ together with a cycle/face-embedding of G. Even though the algorithm works for a set of disjoint cycles, we discuss the situation for one cycle here. This construction can be easily extended to handle a set of disjoint cycles at once.

Analogously to Klau's work [3] we substitute the cycle C by a box (that is represented again by a new cycle C'): We substitute each cycle vertex v by a path $(v_1, \ldots, v_{\delta_v})$ where δ is the number of edges incident to v leaving the cycle. Each vertex v_i has degree 3 and represents one outgoing edge. The order of the path is given by the order of the outgoing edges according to the embedding.

The set of vertices v_i, $v \in C$, forms a new cycle of degree 3 vertices. An original cycle vertex with no edge leaving C does not have a corresponding substitute in C'. We will deal with the re-transformation of such vertices later.

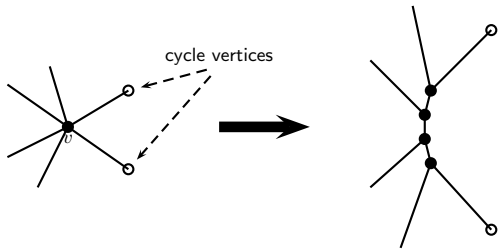


Fig. 2. Each cycle vertex v is split such that each outgoing edge corresponds to one copy. The order is maintained.

The interior face of the new cycle C' must have no angle greater than 180 degrees at its vertices and no angle of 270 degrees in between. This can be achieved similar to Klau's adaptions [3]. With these two constraints we make sure that the interior face has the shape of a rectangle.

We further want all outgoing edges of one original vertex to lie on one side of this rectangle. Thus, between all vertices v_i of the new cycle that correspond to the same original vertex v there must be no bend. Furthermore, there must be no bend of cycle edges at a vertex v_i if it is not the first or last vertex of the corresponding original vertex v ($i \notin \{1, \delta_v\}$).

Additionally, we force the rectangle formed by C' to have large enough sides such that the original cycle C can be drawn circularly in its interior. If n_C is the number of vertices in C, we let $\alpha = 2 \cdot \pi / n_C$ and define its minimal radius $r_{min} = 1/\sqrt{(\sin\alpha)^2 + (1 - \cos\alpha)^2}$. Each side of the rectangle should be at least $2 \cdot (r_{min} + 1)$ long.

As a final step we replace the rectangle by a circle to represent the original cycle using the predefined radius and the center of the rectangle as the center of the circle. We make sure that no edge leaving the cycle crosses a circle edge.

Let v be an original vertex that we want to place on the circle and let $v_1, \ldots, v_{\delta_v}$ be its corresponding vertices on the rectangle boundary. All v_i lie on one side of the boundary as discussed earlier. We place tangents from v_1 and v_{δ_v} on the circle. The interior segment of the circle is the allowed area where we may place v such that no outgoing edge (which connects v with some v_i) crosses the circle.

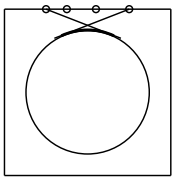


Fig. 3. The eligible circle segment for the position of the cycle vertex is the bold segment.

Let M be the center of the circle. We define lines from v_1 and v_{δ_v} to M. The crossing point of the bisector of these lines with the circle is our position for v. If v does not lie in the previously calculated eligible segment we move it to its nearest eligible point.

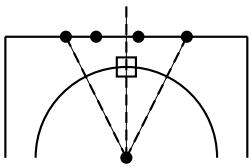


Fig. 4. The new position of the cycle vertex is marked by the square. It is the crossing point of the cycle and the bisector.

Finally, v is connected to each v_i. It is now obvious that these edges do not cross each other and do not cross the circle in any point except for v.

All cycle vertices of degree 2 have not been positioned yet. They will be placed on the circle in mid-distance between their neighbors.

In a clean-up-step we remove the rectangle edges and substitute the rectangle vertices (which have degree 2 now) with bends.

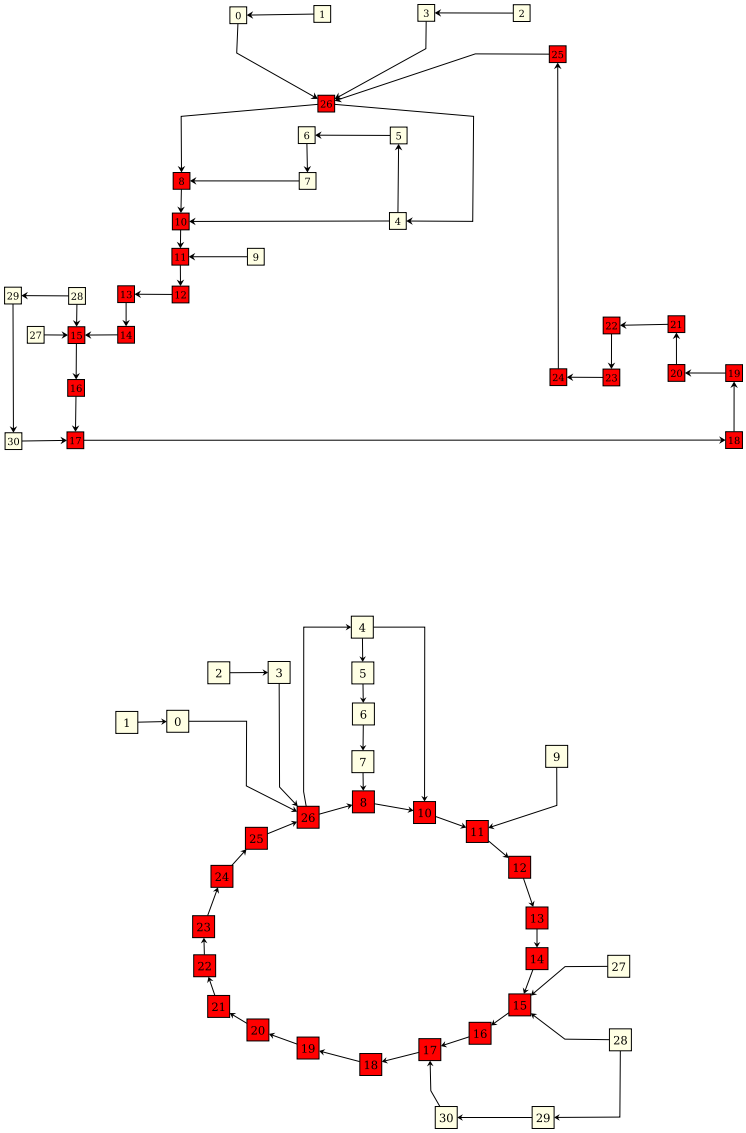


Fig. 5. Example of a graph with a given cycle. The first picture shows a drawing using an ordinary orthogonal layout algorithm while the second picture shows the result of the presented algorithm.

2.3 Preprocessing of non-planar graphs

The presented algorithm (see Section 2.2) uses a special class of planar graphs as input. In this section we present a way to deal with graphs that do not belong to this class. Using a pre-processing step, the algorithm described above can accept arbitrary graphs as input.

We give a sketch of a well-known crossing minimization method that transforms a non-planar graph into a planar graph by substituting crossings into dummy vertices. Then we present our adaption of this algorithm to make sure that a given cycle in the graph will not be crossed by any other edge. The aim is to minimize the number of dummy vertices (e.g. crossings) under the constraint that cycle edges may not be involved in any crossing.

Starting with the empty subgraph $(V, \emptyset)$ of some graph $G = (V, E)$, the incremental method tries to add one edge from E after the other. Whenever adding an edge would destroy planarity, it is discarded, otherwise it is added permanently to the subgraph being constructed. The result is a maximal planar subgraph of G, that, however, is not a maximum planar subgraph in general.

After calculating a maximal planar subgraph, the discarded edges have to be reinserted. Our objective is to insert them one by one such that the minimum number of crossings is produced for each edge. This can be done by inserting an edge optimally over all planar embeddings of the planar subgraph as described in [2]. In each step the crossings are substituted by dummy vertices such that the graph remains planar.

Changes In order to make sure that no edge of the given cycle is crossed by another edge, we make slight changes to the presented algorithm. First, we transform the given cycle into a wheel as explained in Theorem 1.

When constructing the maximal planar subgraph of graph G we do not start with the empty edge set but start with the constructed wheel. Notice that the constructed maximal planar subgraph contains the wheel and, as it is planar, no wheel edge is crossed by another edge. When reinserting the other edges we must make sure that no wheel edge is crossed. This is done in a similar way as the edge re-insertion in [1].

The result is a planar graph with dummy vertices representing real crossings such that no edge of the wheel is crossed. By Theorem 1, removing the spokes and shrinking egdes on the rim yields a cycle/face-planar graph.

This adaption, though mentioned here for a single cycle, can easily be applied to instances with a set of disjoint cycles. Thus, the presented drawing algorithm can be applied to a general graph, even in the extended version with more than one cycle.

2.4 Non-disjoint cycles

The described algorithm can be extended to the more general task when a set of cycles is given, where each pair of cycles shares at most one vertex. The

graph must be cycle/face-planar for the set of cycles, of course. However, in this situation, we cannot guarantee all cycles to be drawn circularly as neighboring cycles may thus intersect, but the cycles can be drawn in a quite circular fashion using bezier curves or ellipses.

In the last step of the drawing algorithm the retransformation of the rectangle to a circle cannot always be done when cycles are allowed to intersect in a vertex as this vertex must lie on the boundary of the rectangle. However, bezier curves can be used to maintain the circular drawing of each cycle inside its predefined rectangle. For the outgoing edges we also use bezier curves for the part inside the rectangle in order to avoid crossings. An example of the extended algorithm is given in Figure 6.

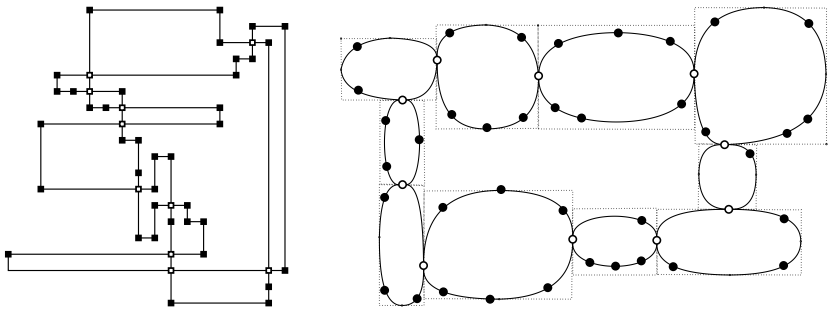


Fig. 6. Example of a set of cycles that can be drawn using the extended algorithm. The left picture shows a drawing using an ordinary orthogonal layout algorithm while the right picture shows the result of the presented algorithm.

Acknowledgment

We thank Dietmar Schomburg and Ralph Schunk for helpful discussions on the application of our approach to biochemical networks.

References

1. C. Buchheim, M. Jünger, A. Menze, and M. Percan. Bimodal crossing minimization. In *Proc. COCOON '06*, volume 4112 of *LNCS*, pages 497–506. Springer-Verlag, 2006.
2. C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005.
3. G. Klau. Quasi-orthogonales Zeichnen planarer Graphen mit wenigen Knicken, 1997.
4. G. Michal. Biochemical pathways, 1993.
5. R.Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.