Bimodal Crossing Minimization[*]

Christoph Buchheim, Michael Jünger, Annette Menze, and Merijam Percan

Universität zu Köln, Institut für Informatik,
Pohligstraße 1, 50969 Köln, Germany
{buchheim, mjuenger, menze, percan}@informatik.uni-koeln.de

Abstract. We consider the problem of drawing a directed graph in two dimensions with a minimum number of crossings such that for every node the incoming edges appear consecutively in the cyclic adjacency lists. We show how to adapt the planarization method and the recently devised exact crossing minimization approach in a simple way. We report experimental results on the increase in the number of crossings involved by this additional restriction on the set of feasible drawings. It turns out that this increase is negligible for most practical instances.

1 Introduction

The importance of automatic graph drawing stems from the fact that many different types of data can be modeled by graphs. In most applications, the interpretation of an edge is asymmetric, so that the graph is intrinsically directed. This is the case, e.g., for metabolic networks. Here, the incoming edges of a reaction node correspond to reactants, while the outgoing edges correspond to products of the modeled reaction. Consequently, a good layout of such a network should separate incoming from outgoing edges, e.g., by letting the incoming edges enter on one side of the node and letting the outgoing edges leave on the opposite side. By this, the human viewer is able to distinguish reactants from products much more easily; see Figure 1.

In spite of its practical relevance, the direction of edges is ignored by many graph drawing algorithms. The graph is processed as an undirected graph first; only after the positions of nodes and edges have been determined the direction is visualized by replacing lines by arrows. An important exception is given by hierarchical drawings, in which incoming and outgoing edges are separated by definition. Furthermore, a polynomial time algorithm for hierarchical drawings of digraphs that allows directed cycles and produces the minimum number of bends is given by Bertolazzi, Di Battista and Didimo in [1]. However, the restriction to this special type of drawing might lead to many more crossings than necessary.

In this paper, our aim is to adapt the planarization method in order to obtain the desired separation of incoming and outgoing edges. We focus on the planarization step itself, i.e., the computation of a planar embedding of the graph after eventually adding virtual nodes representing edge crossings. The objective

[*] Partially supported by the Marie Curie RTN ADONET 504438 funded by the EU.

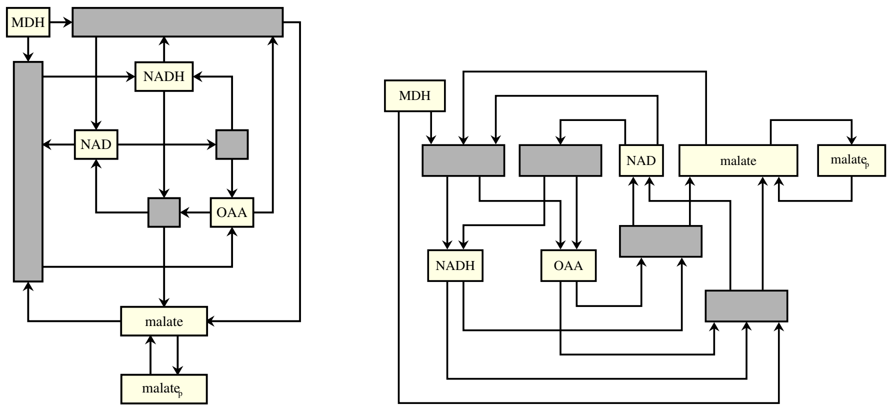


Fig. 1. Two drawings of the same graph, both have three crossings, with unsorted (left) and sorted (right) incoming and outgoing edges; gray nodes represent reactions

is to add as few such nodes as possible. For a comprehensive survey over the planarization approach, see [11].

In order to obtain the separation of edges, we consider the additional bimodal restriction that all incoming edges appear consecutively in all cyclic adjacency lists. We show how to adapt the well-known approach based on finding a planar subgraph first and then reinserting the missing edges one after the other in a very efficient way. We use an experimental evaluation to investigate the question of how many additional crossings have to be expected from restricting the class of feasible embeddings in this way. The results show that—for practical instances—this increase is usually negligible.

We do not address the question of how to realize the resulting embedding by an actual drawing of the graph. Notice however that once we have such an embedding at hand, it is easily possible to adapt, e.g., the orthogonal layout algorithm such that incoming and outgoing edges lie on opposite sides [12].

In Section 2 we recall the concept of bimodality and describe the basic transformation used by the evaluated algorithms. Next we propose a postprocessing technique that can be combined with any crossing reduction approach, see Section 3. Then we look into the planarization method; the problem of finding a planar subgraph is considered in Section 4, while edge reinsertion is dealt with in Section 5. In Section 6, we discuss a recently developed exact approach for crossing minimization. In Section 7, we present an experimental evaluation showing that the number of crossings computed by different methods does not grow much by our additional requirement. Section 8 summarizes the results.

2 Bimodal Embeddings

An embedding of a graph $G = (V, E)$ is called *bimodal* if and only if for every vertex v of G the circular list of the edges around v is partitioned into two (possibly empty) linear lists of edges, one consisting of the incoming edges and the other consisting if the outgoing edges. A planar digraph is *bimodally planar* if and only if it has a bimodal embedding that is planar. This structure was first investigated by Bertolazzi, Di Battista, and Didimo in [1]. Bimodal planarity of a graph G can be decided by testing planarity of a simple transformation of G in $\mathcal{O}(|V|)$ time [1]. The transformation is applied in the following way: for every node v of G expand v by an expansion edge e and add all incoming edges of v to one end-node v_- of e and all outgoing edges to the other end-node v_+ of e. The resulting graph is denoted by $G_d = (V_d,E_d)$ in the following. We call G_d the *d-graph* of G. An illustration of this construction is given in Figure 2.

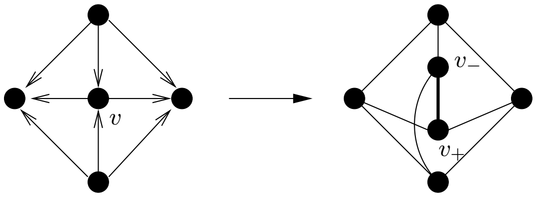


Fig. 2. A directed graph G and its d-graph G_d. The bold edge is an expansion edge. Note that G_d is equal to $K_{3,3}$.

Throughout this paper, we will denote the set of all expansion edges by E'. We use this simple transformation for adapting techniques for undirected crossing minimization to the directed variant. Planar directed graphs are not necessarily bimodally planar. By Kuratowski's theorem, this can only happen if a $K_{3,3}$ or K_5 subdivision is created by the transformation into a d-graph. Note that for graphs with all nodes of degree at most three the transformation of G to G_d is trivial, as no nodes are split in this case. In particular, this holds for cubic graphs that are defined by the property that all nodes have degree three. Therefore, a directed cubic graph is bimodally planar if and only if it is planar. This is also true for graphs in which each node has at most one incoming edge or at most one outgoing edge.

3 Naive Post-processing Approach

We first discuss a post-processing procedure that can be used after applying any crossing reduction algorithm or heuristic to the d-graph G_d. Our aim is to embed G_d such that no expansion edge crosses any other edge; contracting expansion edges then yields an embedding of G with the desired separation of incoming and outgoing edges. So assume that any embedding of G_d is given. We first delete all edges crossing any expansion edge. If two expansion edges cross each other, we delete one of them. Next, we reinsert all deleted edges one after another, starting with the deleted expansion edges. As explained in Section 5.1 below, we can insert a single edge with a minimal number of crossings for the fixed embedding computed so far such that crossings with expansion edges are prevented. If reinserting an expansion edge produces any crossings, the crossed (non-expansion) edges have to be deleted and put to the end of the queue of edges to be reinserted. At the end of this reinsertion process, no expansion edge will cross any other edge. However, the number of crossings of the remaining edges might grow significantly in this approach. In the following sections, we explain how to get better results by adapting well-known crossing minimization approaches for our purposes, especially the planarization approach.

4 Maximum Bimodally Planar Subgraphs

It is well-known that the maximum planar subgraph problem—the problem of finding a planar subgraph of a given graph that contains a maximum number of edges—is NP-hard. Recently, it was shown that this remains true even for cubic graphs:

Theorem 1 (Faria et al. [4]). *The maximum planar subgraph problem is NP-hard for cubic graphs.*

As a cubic graph is equal to its d-graph, we derive that this also holds for the maximum bimodally planar subgraph problem:

Corollary 1. *It is an NP-hard problem to compute a maximum bimodally planar subgraph of a directed graph, even for a cubic graph.*

For computing *maximal* bimodally planar subgraphs, i.e., bimodally planar subgraphs such that adding any further edge of G destroys bimodally planarity, we do the following: it is easy to see that the bimodally planar subgraphs of G are in one-to-one correspondence to the planar subgraphs of G_d containing all expansion edges. Thus we have to modify a given maximal planar subgraph algorithm such that it never deletes any expansion edge. Methods for finding maximal planar subgraphs have been studied intensively [8,10,3]; here we only discuss the incremental method; see Section 4.1. We also have a look at the exact approach; see Section 4.2.

4.1 Incremental Method

Starting with the empty subgraph $(V_H, \emptyset)$ of some graph $H = (V_H, E_H)$, the incremental method tries to add one edge from E_H 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 H, which however is not a maximum planar subgraph in general. To find a maximal bimodally planar subgraph of G, we have to compute a maximal planar subgraph of its d-graph G_d. However, this subgraph must always contain all expansion edges, so that the latter can be contracted at the end. We thus have to start with the subgraph (V_d, E')—which is obviously planar—instead of the empty subgraph $(V_d, \emptyset)$. Then we try to add the remaining edges $E_d \setminus E'$ as before. The resulting subgraph of G_d corresponds to a maximal bimodally planar subgraph H of G.

4.2 Exact Method

An exact approach for finding a maximum planar subgraph of $H = (V_H, E_H)$ based on polyhedral techniques was devised in [9]. The problem is modeled by an integer linear program (ILP) as follows: for every edge $e \in E_H$, a binary variable x_e is introduced, having value one if and only if e belongs to the chosen subgraph. To enforce that the modeled subgraph is planar, one has to make sure that it contains no Kuratowski subgraph of H, i.e., no subdivision of K_5 or $K_{3,3}$. In terms of the model, this is equivalent to the constraint $\sum_{e \in K} x_e \leq |K| - 1$ for every (edge set of a) Kuratowski graph K in G. As we search for a planar subgraph containing the maximal number of edges, the number of variables set to one should be maximized. The integer linear program is thus

$$\begin{aligned} \max \ & \textstyle\sum_{e \in E_H} x_e \\ \text{s.t.} \ & \textstyle\sum_{e \in K} x_e \leq |K| - 1 && \text{for all Kuratowski subgraphs } K \text{ of } G \\ & x_e \in \{0, 1\} && \text{for all } e \in E_H \ . \end{aligned}$$

This ILP can now be solved by branch-and-cut. However, in order to improve the runtime of such algorithms and hence obtain a practical solution method, one has to further investigate this formulation and exhibit other classes of valid inequalities as well as fast techniques for finding violated constraints for a given fractional solution. For details, the reader is referred to [9]. This solution approach can easily be adapted to our situation: we have to ensure that the edges in E' always belong to the chosen subgraph, i.e., we have to add the constraint $x_e = 1$ to the ILP, for each expansion edge $e \in E'$. Observe that this type of constraint is harmless with respect to the complexity of the problem, as it cuts out a face from the polytope spanned by the feasible solutions of the ILP.

5 Edge Reinsertion

After calculating a maximal (resp., maximum) bimodally planar subgraph, the deleted edges have to be reinserted. Our objective is to reinsert them one by one so that the minimum number of crossings are produced for each edge. This can be done in two different ways: either by inserting an edge into a fixed bimodally planar embedding of the bimodally planar subgraph, see Section 5.1, or by inserting an edge optimally over all bimodally planar embeddings of the bimodally planar subgraph, see Section 5.2. Again, we have to treat expansion edges differently, as they may not be involved in any edge crossings.

5.1 Fixed Embedding

Given a fixed embedding $\Gamma(G_d)$ of G_d, it is easy to insert an edge $e(v, w)$ into $\Gamma(G_d)$ such that a minimal number of crossings is produced. For this, one can use the *extended dual graph* D of $\Gamma(G_d)$, the nodes of which are the faces of $\Gamma(G_d)$ plus two nodes v_D and w_D corresponding to v and w. For each edge in $E_d \setminus E'$, we have the dual edge in D. Additionally, we connect v (resp., w) with all nodes in D corresponding to faces that are adjacent to v (resp., w) in $\Gamma(G_d)$. Then we calculate the shortest path from v to w in the extended dual graph and insert the edge e into $\Gamma(G_d)$ along this path, replacing crossings by dummy nodes. Clearly, the shortest path does not cross any edge of E' as its dual edge is not included in D. This can be done in $\mathcal{O}(|V|)$ time.

5.2 All Embeddings

In the previous section we have considered reinserting an edge into a fixed embedding. For getting fewer edge crossings, a powerful method is to calculate the shortest path between two nodes v and w over all embeddings. In [6] a linear time algorithm is presented for finding an optimal embedding which allows to insert e with the minimum number of crossings. It uses the SPQR-tree and BC-tree data-structures for representing all planar embeddings of a connected graph. In the same straightforward way as explained in the previous section, this approach can be adapted such that no expansion edge is crossed by any reinserted edge. The resulting algorithm runs in $\mathcal{O}(|V|)$ time.

6 Exact Bimodal Crossing Minimization

It is a well-known fact that the general crossing minimization problem for undirected graphs is NP-hard [5]. More recent results show that this is even true for graphs with all nodes of degree three:

Theorem 2 (Hliněný [7], Pelsmajer, Schaefer, Štefankovič [13]). *The crossing minimization problem is NP-hard for cubic graphs.*

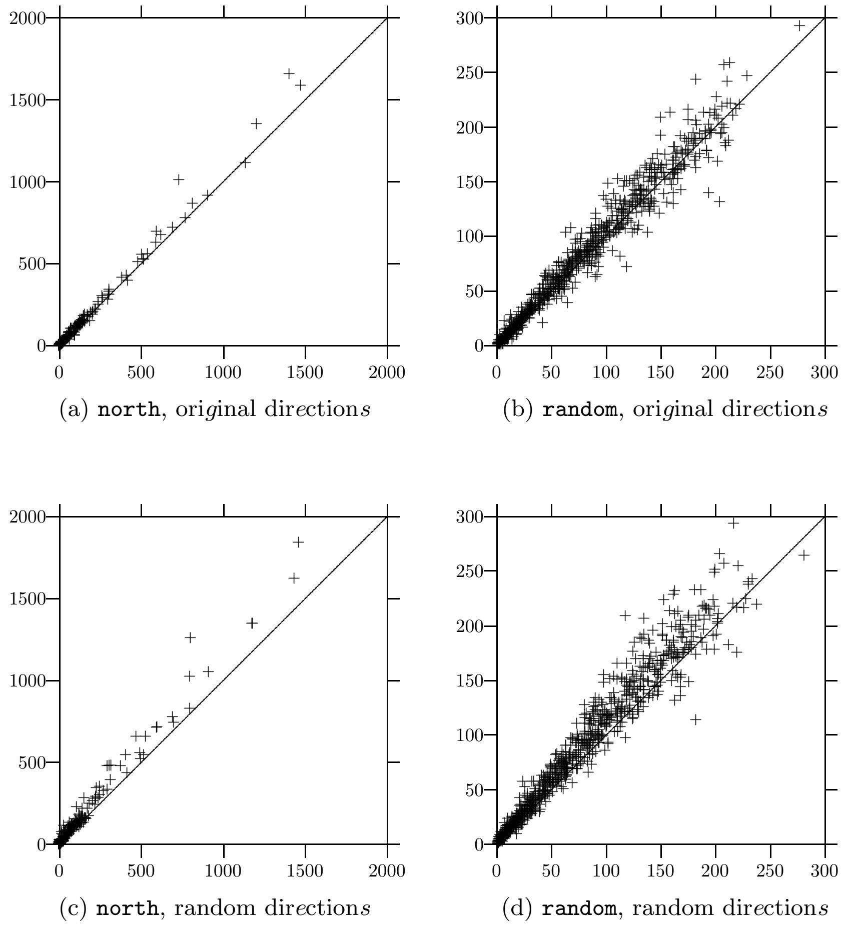
Corollary 2. *It is an NP-hard problem to compute a drawing of G separating incoming and outgoing edges such that the number of crossings is minimal. This even holds for cubic graphs.*

Despite the NP-hardness of undirected crossing minimization, an exact approach has been devised recently [2]; a branch-and-cut algorithm is proposed for minimizing the number of crossings over all possible drawings. The first step in this approach is to replace every edge of the graph by a path of length (at most) $|E|$. After this, one may assume that every edge has a crossing with at most one other edge. The ILP model used in this approach contains a variable x_{ef} for all pairs of edges $(e, f) \in E \times E$, having value one if and only if there is a crossing between e and f in the drawing to be computed. By appropriate linear constraints, one can ensure that the given solution is realizable, i.e., corresponds to some drawing of G. Again, it is easy to adjust this method to our problem, i.e., the problem of computing a crossing-minimal drawing with incoming and outgoing edges separated. For this, we can apply the above algorithm to the graph G_d. Then we only have to make sure that the expansion edges do not have any crossings in the computed solution. We can thus do the adjustment as follows: first observe that the edges in E' do not have to be replaced by a path at all, as they are not allowed to produce crossings. Now we can just omit the variable x_{ef} whenever $e \in E'$ or $f \in E'$, and thereby set this variable to zero implicitly. The resulting ILP will thus have exactly the same number of variables as the original ILP for the non-transformed graph. It will not become harder structurally, as it arises from setting variables to zero.

7 Experimental Comparison

In the previous sections, we showed how to adapt several crossing minimization algorithms and heuristics in a simple way such that for directed graphs the sets of incoming and outgoing edges are separated in the adjacency lists. This is obtained by transforming the original directed graph into a new undirected graph where certain edges do not allow any crossings. From the nature of this transformation and the described modifications, it is obvious that the runtime is not affected negatively. We also observed this in our experiments. For this reason, we focused on the number of crossings in the evaluation reported in the following: we are interested in comparing the number of crossings when (a) the direction of edges is ignored, i.e., crossing minimization is done as usual, and (b) we apply the transformation in order to separate incoming from outgoing edges. Theoretically, the crossing number cannot decrease by our modification, but it is possible that it grows considerably. However, our experiments show that for practical graphs the number of crossings is not increased significantly. In fact, the increase in the number of crossings is marginal compared with the variance due to the randomness of the heuristics, such that for many instances the number of crossings after the transformation even decreases. Combining this observation with the simpleness of implementation and the fact that runtime does not increase, our claim is that these techniques should always be applied when dealing with (meaningfully) directed edges. For the experiments, we used the instances of the Rome library of directed graphs [14], consisting of two sets of graphs called `north` and `random`. The former contains 1277 directed acyclic graphs on 10 to 100 nodes derived from real-world instances. The latter contains 909 directed acyclic graphs randomly generated in a specific way, they are much denser in general. We first applied the simple incremental method (Section 4.1) combined with the optimal edge reinsertion over all embeddings (Section 5.2). As mentioned above, it turned out that the increase in the number of crossings when separating incoming and outgoing edges is very small in general. This is shown in Figure 3 (a) and (b), where each instance is given by a plus sign. Its x-coordinate is the number of crossings before the transformation and the y-coordinate is the number of crossings afterwards. In particular, each cross on the diagonal line represents an instance with the same number of crossings before and afterwards. A cross above the diagonal represents an instance for which the number of crossings increases. Due to the randomness of the heuristics, there are also crosses below the diagonal, in particular for the `random` instances.

Another interesting finding is the negligible increase in the number of crossings for planar graphs: if G is planar, then G_d is not necessarily planar. Anyway, if



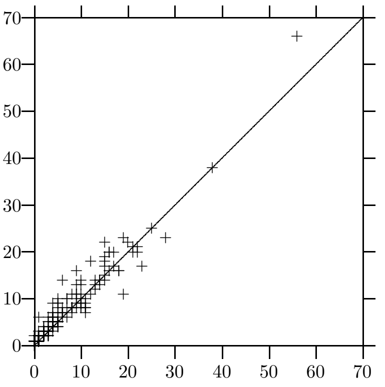
(a) `north`, original directions (b) `random`, original directions

(c) `north`, random directions (d) `random`, random directions

Fig. 3. Numbers of crossings before and after the transformation, using the incremental planar subgraph heuristic. For non-planar graphs, the average increase is 0.36 % (a), 0.59 % (b), 0.95 % (c), and 0.87 % (d), respectively.

we consider all 854 planar `north` instances, then the average number of crossings after the transformation is only 0.04, i.e., in most cases the graph remains planar. The set of `random` instances does not contain any planar graph. We next applied the optimal planar subgraph method (Section 4.2), again in combination with the optimal edge reinsertion over all embeddings (Section 5.2). As many instances could not be solved within a reasonable running time, we had to set a time limit of five CPU minutes (on an Athlon processor with 2.0 GHz). Within this time limit, 89 % of the `north` instances and 33 % of the `random` instances could be solved. The results are shown in Figure 4; the general picture is similar to the one for the incremental method.

The directed graphs contained in the libraries `north` and `random` are all acyclic. This fact might favor a small number of additional crossings. For this


(a) `north`, original directions

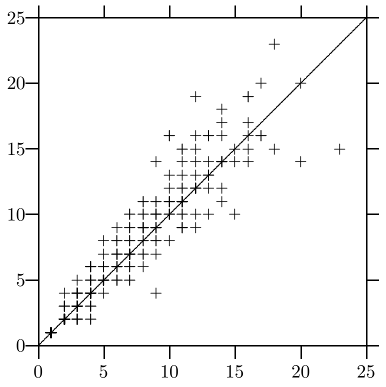

(b) `random`, original directions

Fig. 4. Numbers of crossings before and after the transformation, using the optimal planar subgraph method. For non-planar graphs, the average increase is 3.30 % (a) and 4.21 % (b), respectively.

reason, we also examined graphs with a random direction for each edge. To allow us to compare the corresponding results to the results presented so far, we used the `north` and `random` instances again, this time with the direction of each edge reversed with a probability of 1/2. The results obtained with the incremental heuristic are displayed in Figure 3 (c) and (d). In fact, the increase in the number of crossings induced by sorting adjacency lists is more obvious now compared to Figure 3 (a) and (b), but it is still very small. Nevertheless, we conjecture that in theory the requirement of separating incoming and outgoing edges may induce a quadratic number of edge crossings even for planar graphs. We have constructed a family of directed planar graphs G_k such that G_k has $O(k)$ edges and such that the planarization heuristic has always produced $\Omega(k^2)$ crossings when separating incoming from outgoing edges. The graph G_k is defined as follows: it consists of two wheel graphs W_{2k} sharing their rim; one of them has all spokes directed from the rim to the hub, the other one has spokes with alternating direction. We applied the planarization method to the graphs G_k many times, with enforced separation of incoming and outgoing edges. For all k, the smallest number of crossings we could find was $\sum_{i=1}^{k} \lfloor i/2 \rfloor = \Theta(k^2)$. We conjecture that this is the minimum number of crossings for all bimodal drawings of G_k. This would mean that a quadratic number of crossings is unavoidable even for planar graphs.

8 Conclusion

We can summarize the statement of this paper as follows: whenever the direction of edges in a graph carries significant information, this should be stressed by separating incoming and outgoing edges in the adjacency lists. We have shown how crossing reduction algorithms can be adapted in order to comply with this requirement. The necessary changes are not only easy to implement but also neutral with respect to runtime. As our experiments show, the number of crossings can be expected to grow only slightly for practical instances.

Acknowledgement

We thank Dietmar Schomburg and Ralph Schunk for helpful discussions on the application of our approach to biochemical networks. Moreover, we are grateful to Maria Kandyba for implementing the presented heuristic methods and to Katrina Riehl and Stefan Hachul for proof-reading this paper.

References

1. P. Bertolazzi, G. Di Battista, and W. Didimo. Quasi-upward planarity. *Algorithmica*, 32:474–506, 2002.
2. C. Buchheim, D. Ebner, M. Jünger, G. W. Klau, P. Mutzel, and R. Weiskircher. Exact Crossing Minimization, in: P. Healy, N.S. Nikolov (eds.), *Graph Drawing 2005*. LNCS 3843, pp. 37–48, 2006.
3. H. N. Djidjev. A linear algorithm for the maximal planar subgraph problem. In *WADS '95*, volume 955 of *LNCS*, pages 369–380. Springer-Verlag, 1995.
4. L. Faria, C. M. H. de Figueiredo, and C. F. X. de Mendonça N. Splitting number is NP-complete. *Discrete Applied Mathematics*, 108(1–2):65–83, 2001.
5. M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983.
6. C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005.
7. P. Hliněný. Crossing number is hard for cubic graphs. In *MCFS 2004*, pages 772–782, 2003.
8. R. Jayakumar, K. Thulasiraman, and M. N. S. Swamy. $O(n^2)$ algorithms for graph planarization. *IEEE Transactions on Computer-Aided Design*, 8:257–267, 1989.
9. M. Jünger and P. Mutzel. Maximum planar subgraphs and nice embeddings: Practical layout tools. *Algorithmica*, 16(1):33–59, 1996.
10. J. A. La Poutré. Alpha-algorithms for incremental planarity testing. In *STOC '94*, pages 706–715, 1994.
11. A. Liebers. Planarizing graphs – a survey and annotated bibliography. *Journal of Graph Algorithms and Applications*, 5(1):1–74, 2001.
12. A. Menze. Darstellung von Nebenmetaboliten in automatisch erzeugten Zeichnungen metabolischer Netzwerke. Master's thesis, Institute of Biochemistry, University of Cologne, June 2004.
13. M. J. Pelsmajer, M. Schaefer, and D. Štefankovič. Crossing number of graphs with rotation systems. Technical report, Department of Computer Science, DePaul University, 2005.
14. Rome library of directed graphs.
http://www.inf.uniroma3.it/people/gdb/wp12/directed-acyclic-1.tar.gz.