Optimal k-Level Planarization and Crossing Minimization

Graeme Gange[1], Peter J. Stuckey[1], and Kim Marriott[2]

[1] Department of Computer Science and Software Engineering
The University of Melbourne, Vic. 3010, Australia
`{ggange,pjs}@csse.unimelb.edu.au`
[2] Clayton School of IT
Monash University, Vic. 3800, Australia
`Kim.Marriott@infotech.monash.edu.au`

Abstract. An important step in laying out hierarchical network diagrams is to order the nodes on each level. The usual approach is to minimize the number of edge crossings. This problem is NP-hard even for two layers when the first layer is fixed. Hence, in practice crossing minimization is performed using heuristics. Another suggested approach is to maximize the planar subgraph, i.e. find the least number of edges to delete to make the graph planar. Again this is performed using heuristics since minimal edge deletion for planarity is NP-hard. We show that using modern SAT and MIP solving approaches we can find *optimal* orderings for minimal crossing or minimal edge deletion for planarization on reasonably sized graphs. These exact approaches provide a benchmark for measuring quality of heuristic crossing minimization and planarization algorithms. Furthermore, we can straightforwardly extend our approach to minimize crossings followed by maximizing planar subgraph or vice versa; these hybrid approaches produce noticeably better layout then either crossing minimization or planarization alone.

1 Introduction

The standard approach for drawing hierarchical network diagrams is a three phase approach in which (a) nodes in the graph are assigned levels producing a k-level graph; (b) nodes are assigned an order so as to minimize edge crossings in the k-level graph; and (c) the edge routes and node positions are computed. There has been considerable research into step (b) which is called *k-level crossing minimization.* Unfortunately this step is NP-hard even for two layers ($k = 2$) where the ordering on one layer is given. Thus, research has focussed on developing heuristics to solve it. In practice the approach is to iterate through the levels, re-ordering the nodes on each level using heuristic techniques such as the barycentric method [1].

An alternative to performing crossing minimization in phase (b) is *k-level planarization* problem. This was introduced by Mutzel [2] and is the problem of finding the minimal set of edges that can be removed which allow the remaining edges in the k-level graph to be drawn without any crossings. Mutzel has argued convincingly that for hierarchical network diagrams with many crossings,

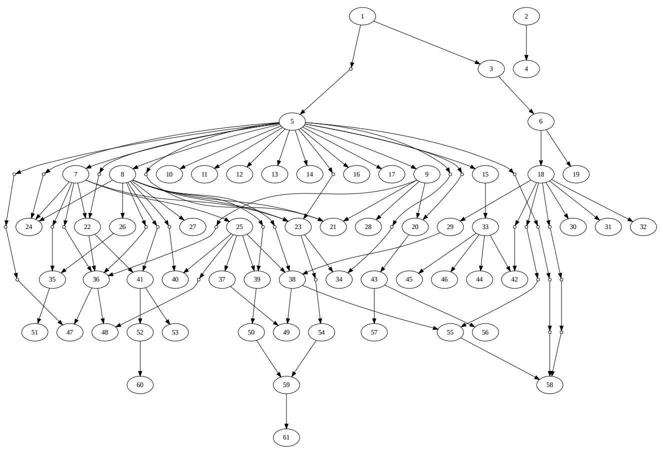


Fig. 1. Graphviz heuristic layout for the *profile* example graph

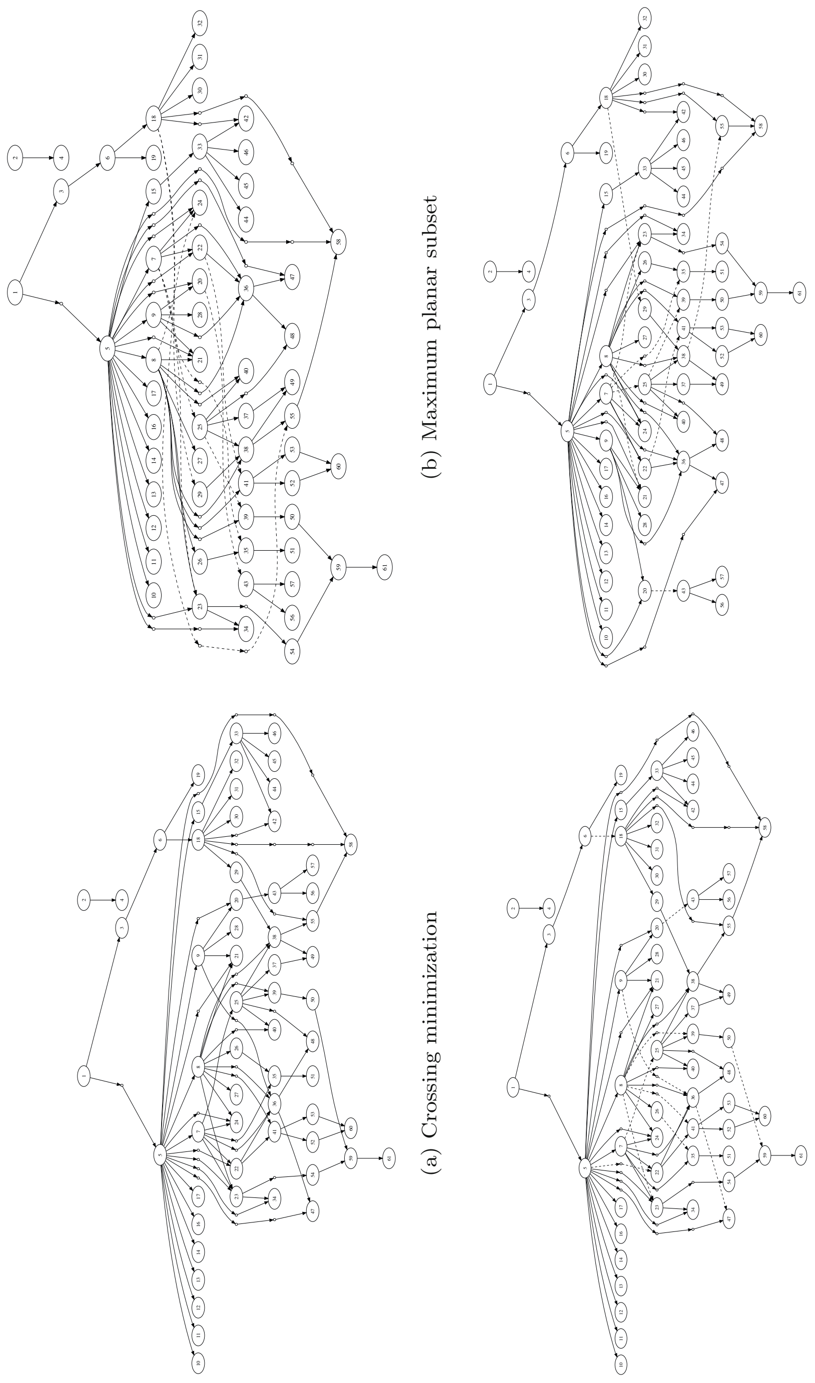
this leads to better drawings than those obtained by simply minimizing the total number of edge crossings. While in some sense simpler than k-level crossing minimization (since the problem is tractable for $k = 2$ with one side fixed) it is still NP-hard for $k > 2$. A disadvantage of k-level planarization is that it does not take into account the number of crossings that the non-planar edges generate and so a poor choice of which edges to remove can give rise to unnecessary edge crossings.

Here we introduce a combination of the two approaches we call *k-level planarization and crossing minimization*. This minimizes the weighted sum of the number of crossings *and* the number of edges that need to be removed to give a planar drawing. We believe that this gives rise to nicer drawings than either k-level planarization or k-level crossing minimization while providing a natural generalization of both.

As some evidence for this consider the drawings shown in Figures 1 and Figure 2 of the example graph `profile` from the GraphViz gallery [3]. Figure 1 shows the layout from GraphViz using its heuristic for edge crossing minimization. It has 54 edge crossing and requires removal of 17 edges to become planar.

The layout resulting from minimizing edge crossings is shown in Figure 2(a). It has 38 crossings, significantly less than the heuristic layout. The layout resulting from maximizing the planar subgraph is shown in Figure 2(b) with deleted edges dotted. It requires only 9 edges to be deleted but has 81 crossings. The layout clearly shows that maximizing the planar subgraph in isolation is not enough, leading to many unnecessary crossings.

The combined model allows us to minimize both crossings and edge deletions for planarity simultaneously. Figure 2(c) shows the result of minimizing crossings and then maximizing the planar subset. It yields 38 crossings and 11 edge deletions. Figure 2(d) shows the results of of maximizing the planar subset and the minimize crossings. It yields 9 edge deletions and 57 edge



(a) Crossing minimization

(b) Maximum planar subset

(c) Minimize crossings, then maximize planar subset

(d) Maximum planar subset, then crossing minimization

Fig. 2. Different layouts of the *profile* example graph

crossings, a substantial improvement over the maximal planar subgraph layout of Figure 2(b).

We believe these combined layouts clearly illustrate that some combination of minimal edge crossing and minimal edge deletions for planarity leads to better layout than either individually. Part of the advantage is simply that displaying edges deleted for planarity differently makes the layout much clearer.

Apart from introducing these combined layouts, the paper has two main technical contributions. The first is to give a binary program for the combined k-level planarization and crossing minimization problem. By appropriate choice of the weighting factor this model reduces to either k-level planarization or k-level crossing minimization. Our basic model is reasonably straightforward but we use some tricks to reduce symmetries, handle leaf nodes in trees and improve bounds for edge cycles.

Our second technical contribution is to evaluate performance of the binary program using both a generic MIP solver and a generic SAT solver. While MIP techniques are not uncommon in graph drawing the use of SAT techniques is quite unusual. Our reason for considering MIP is that MIP is well suited to combinatorial optimization problems in which the linear relaxation of the problem is close to the original problem. However this does not seem true for k-level planarization and/or k-level crossing minimization. Hence it is worth investigating the use of other generic optimization techniques. Over the last decade there has been considerable improvement in SAT solving techniques and they are now capable of solving problems with thousands of variables in a few seconds. Part of this improvement arises by learning combinations of assignments to the Boolean variables that lead to unsatisfiability (called "no goods") as the search for the optimum solution proceeds. The no goods are used to prune the search space leading to orders of magnitude performance improvement.

We find that modern SAT solving with learning, and modern MIP solvers (which now have special routines to handle SAT style models) are able to handle the k-level planarization and crossing minimization problems and their combination for quite large k, meaning that we can solve step (b) to optimality. They are fast enough to find the optimal ordering of nodes on all layers for graphs with hundreds of nodes in a few seconds, so long as the graph is reasonably narrow (less than 10 nodes on each level) and for larger graphs they find reasonable solutions within one minute.

The significance of our research is twofold. First it provides a benchmark for measuring the quality of heuristic methods for solving k-level crossing minimization and/or k-level planarization. Second, the method is practical for small to medium graphs and leads to significantly fewer edge crossings involving fewer edges than is obtained with the standard heuristic approaches. As computers increase in speed and SAT solving and MIP solving techniques continue to improve we predict that optimal solution techniques based on MIP and SAT will replace the use of heuristics for step (b) in layout of hierarchical networks.

Furthermore, our research provides support for the use of generic optimization techniques for exploring different aesthetic criteria. The use of generic techniques allows easy experimentation with, for instance, our hybrid objective function. As another example rather than k-level planarization we might wish to minimize the total number of edges involved in crossings. This is simple to do with generic optimization. Another advantage of generic optimization techniques is that they also readily handle additional constraints on the layout, such as placing some nodes on the outside or clustering nodes together.

The task of reducing crossings in k-layered graphs has received considerable attention, particularly due to the layout algorithm of Sugiyama [4]. However, most of these approaches, such as [5] tend to use heuristics, rather than finding the global optimum.

The most closely related work is on the use of MIP and branch-and-bound techniques for solving k-level crossing minimization. Jünger and Mutzel [6] compared heuristic methods for two layer crossing minimization with a MIP encoding solved using a specialized branch-and-cut algorithm to solve to optimality. They found that the MIP encoding for the case when one layer is fixed is practical for reasonably sized graphs. In another paper, Jünger *et al* [7] gave a 0-1 model for k-level crossing minimization and solved it using a generic MIP solver. They found that at that time MIP techniques were impractical except for quite small graphs. We differ from this in considering planarization as well and in investigating SAT solvers. Randerath *et al* [8] gave a partial-MAXSAT model of crossing minimization, however did not provide any experiments. We show that SAT solving with learning, and more recent MIP solvers (which now have special routines to handle SAT style models) are now practical for reasonably sized graphs.

Also related is Mutzel [2] which describes the results of using a MIP encoding with branch-and-cut for the 2-level planarization problem. Here we give a binary program model for k-level planarization and show that SAT with learning and modern MIP solvers can solve the k-level planarization problem for quite large k. We use a similar model to that of Jünger and Mutzel but examine both MIP and SAT techniques to solving it.

The paper is organized as follows. In the next section we give our model for combined planarity and crossing minimization. In Section 3 we show how to improve the model by taking into account graph properties. In Section 4 we give results of experiments comparing the different measures, and finally in Section 5 we conclude.

2 Model

A general framework for generating layouts of hierarchical data was presented by [4]. This proceeds in three stages. First, the vertices of the graph are partitioned into horizontal layers. Then, the ordering of vertices within these horizontal layers is permuted to reduce the number of edge crossings. Finally, these layers are positioned to straighten long edges and minimize edge length. Our focus is on the second stage of this process – permuting the vertices on each layer.

Consider a graph with nodes divided into k layers, with edges restricted to adjacent layers, ie. edges from layer i to $i+1$. Denote the nodes in the $k-th$

layer by $nodes[k]$, and the edges from layer k to layer $k+1$ by $edges[k]$. For a given edge e, denote the start and end nodes by $e.s$ and $e.d$ respectively.

The combined model for maximal planar subgraph and crossing minimization is defined by the binary program:

$$\text{min } \sum_{k \in levels} C \sum_{e,f \in edges[k]} c_{(e,f)} + P \sum_{e \in edges[k]} r_e \tag{1}$$

s.t.

$$\bigwedge_{k \in levels} \bigwedge_{i,j,k \in nodes[k]} l_{(i,j)} \wedge l_{(j,k)} \rightarrow l_{(i,k)} \tag{2}$$

$$\bigwedge_{k \in levels} \bigwedge_{e,f \in edges[k]} c_{(e,f)} \leftrightarrow l_{(e.s,f.s)} \oplus l_{(e.d,f.d)} \tag{3}$$

$$\bigwedge_{k \in levels} \bigwedge_{e,f \in edges[k]} r_e \vee r_f \vee \neg c_{(e.f)} \tag{4}$$

The variable $l_{(i,j)}$ indicates node i is before j in the level ordering. The variable $c_{(e,f)}$ indicates that edge e crosses edge f. The variable r_e indicates that edge e is deleted to make the graph planar. The constants C and P define the relative weights of crossing minimization and edge deletion for planarity. The 3-cycle constraints of Equation 2 ensures that the order variables are assigned to a consistent ordering. Equation 3 defines the edge crossings variables in terms of the ordering: the edges cross if the relative order of the start and end nodes are reversed. It is encoded in clauses as

$$c_{(e,f)} \vee l_{(e.s,f.s)} \vee \neg l_{(e.d,f.d)}, \quad c_{(e,f)} \vee \neg l_{(e.s,f.s)} \vee l_{(e.d,f.d)},$$
$$\neg c_{(e,f)} \vee l_{(e.s,f.s)} \vee l_{(e.d,f.d)}, \neg c_{(e,f)} \vee \neg l_{(e.s,f.s)} \vee \neg l_{(e.d,f.d)}.$$

The planarity requirement is encoded in Eq. 4 which states that for each pair either one is removed, or they don't cross. The combined model uses $O(k.(e^2 + n^2))$ Boolean variables and is $O(k.(n^3 + e^2))$ in size.

We can convert this clausal model to a MIP binary program by converting each clause $b_1 \vee \cdots b_l \vee \neg b_{l+1} \vee \cdots \vee \neg b_m$ to the linear constraint $b_1 + \cdots + b_l - b_{l-1} - \cdots - b_m \geq m - l + 1$.

Long edges are handled by adding intermediate nodes in the levels that the long edges cross and breaking the edge into components. For crossing minimization each of these new edges is treated like an original edge. For the minimal deletion of edges each component edge in a long edge e is encoded using the same deletion variable r_e.

By adjusting the relative weights for crossing C, and planarization P, we can create and evaluate new measures of clarity of the graph. With $C = 1 + \sum_{k \in levels} |edges[k]|$ and $P = 1$ we first minimize crossings, then minimize edge deletions for planarity. With $C = 1$ and $P = \sum_{k \in levels} |edges[k]|^2$ we first minimize edge deletions and then crossings.

3 Additional Constraints

While the basic model described in Section 2 are sufficient to ensure correctness, finding the optimum still requires a great deal of search. We can modify the model to significantly improve performance.

First note that we add symmetry breaking by fixing the order of the first two nodes appearing on the same level. If the graph to be layed out has more

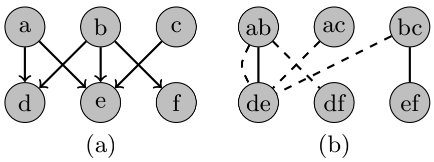


Fig. 3. (a) A graph, with an initial ordering (b) The corresponding vertex-exchange graph

symmetries than this left-to-right symmetry we could use this to fix more variables (although we don't do this in the experiments). Next, we can improve edge crossing minimization by using as an upper bound the number of crossings in a heuristic layout. We could also use heuristic solutions to bound planarity but doing so requires computing how many edges need deletion, which is non-trivial.

3.1 Cycle Parity

Healy and Kuusik introduced the vertex-exchange graph [9] for analyzing layered graphs. Each edge in the vertex-exchange graph corresponds to a potential crossing in the initial graph; each node corresponds to a pair of nodes within a level.

Consider the graph shown in Figure 3(a), its vertex-exchange graph is shown in Figure 3(b). Note there are two edges (ab, de) corresponding to the two pairs $((a, d), (b, e))$ and $((a, e), (b, d))$. Edges corresponding to crossings in Figure 3(a) are shown as solid, the rest are dashed.

For any given cycle in the vertex exchange graph, permuting nodes within a layer will maintain parity in the number of crossings in the cycle. For cycles with an odd number of crossings, this means that at least one of the pairs of edges in the cycle will be crossing. This can be represented by the clause $\bigvee_{(e,f)\in cycle} c_{(e,f)}$. When finding the maximal planar subgraph, we then know that at least one edge involved in the cycle must be removed from the subgraph. Similarly since the cycle is even in length we know that not all edges can cross, represented by $\bigvee_{(e,f)\in cycle} \neg c_{(e,f)}$. Both these constraints can be added to the model.

A special case of cycle parity is the $K_{2,2}$ subgraph. This subgraph always produces exactly one crossing, irrespective of the relative orderings of the nodes in the subgraph. When minimizing crossings, the corresponding $c_{(e,f)}$ variables need not be included in the objective function, which considerably simplifies the problem structure. Note that, for example, a $K_{3,3}$ subgraph contains 9 $K_{2,2}$ subgraphs, and each of the 9 $c_{e,f}$ variables arising can be ommitted from the problem. For the experiments we add constraints for cycles of length 6 or less, since the larger cycles did not improve performance.

3.2 Leaves

It is not difficult to prove that if a node on layer k has m child leaf nodes (unconnected to any other node) on layer $k + 1$, then all of these leaf nodes can be ordered together.

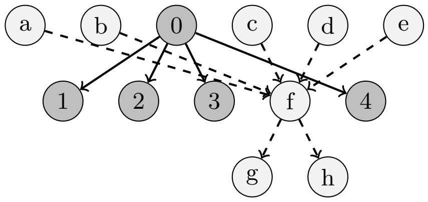


Fig. 4. A partial layout with respect to some leaf nodes 1,2,3,4

Consider the partial layout illustrated in Figure 4, where each node 1,2,3 and 4 is a leaf node with no outgoing arcs. If we place a node f in between nodes 1,2,3 and 4 (as illustrated) there is always at least as good a crossing solution by placing f either before or after all of them. Here since there are 2 parents before 0 and 3 after, f should be placed after 4, leading to 8 crossings rather than the 9 illustrated.

Similarly maximizing planarity always requires that all edges to siblings left of f be removed or all edges from parents before 0, and all edges to siblings right of f or all edges from parents after 0. An optimal solution always results by either deleting all edges to leaf nodes (which makes the leaf positions irrelevant), or ordering f after all leaves and deleting all edges from parents before 0, or ordering f before all leaves and deleting all edges from parents after 0.

Since there is no benefit in splitting leaf siblings we can treat them as a single node, but note we must appropriately weight the edge resulting, since it represents multiple crossings and multiple edge deletions.

Let N be a set of m leaf nodes from a single parent node i. We replace N by a new node j', and replace all edges $\{(i,j) \mid j \in N\}$ by the single edge (i,j'). We replace each m terms $c_{((i,j),f)}, j \in N$ in the objective function by one term $m \times c_{((i,j'),f)}$ and replace each of the m terms $r_{(i,j)}, j \in N$ in the objective by the term $m \times r_{(i,j')}$.

4 Experimental Results

We tested the binary model on a variety of graphs, using the pseudo-Boolean constraint solver MiniSAT+[10], and the Mixed Integer Programming solver CPLEX 12.0. All experiments were performed on a 3.0GHz Xeon X5472 with 32 Gb of RAM running Debian GNU/Linux 4.0. We ran for a maximum of 60s, and all times are given in seconds. We compared 4 different objective functions:

- crossing minimization: $C = 1$, $P = 0$;
- maximal planar subgraph $C = 0$, $P = 1$;
- crossing minimization then maximal planar subgraph $C = 1 + \sum_{k \in levels} |edges[k]|$, $P = 1$; and
- maximal planar subgraph then minimize crossings $C = 1$, $P = \sum_{k \in levels} |edges[k]|^2$.

Table 1. Time to find and prove the minimal crossing layout and maximal planar subgraph for Graphviz examples using MIP and SAT

Problem	Crossing minimization					Maximal planar subgraph			
	graphviz	MIP		SAT		MIP		SAT	
	best	best	solved	best	solved	best	solved	best	solved
crazy	3	**2**	0.04	**2**	**0.01**	**1**	0.03	**1**	**0.01**
datastruct	**2**	**2**	**0.00**	**2**	**0.00**	**1**	0.02	**1**	**0.00**
fsm	**0**	**0**	**0.00**	**0**	**0.00**	**0**	**0.00**	**0**	**0.00**
lion_share	7	**4**	**0.04**	**4**	0.11	**2**	0.05	**2**	**0.02**
profile	54	**38**	**6.81**	54	—	12	—	**9**	**5.39**
switch	**20**	**20**	0.75	**20**	**0.64**	**17**	—	**17**	**34.49**
traffic_lights	**0**	**0**	**0.00**	**0**	**0.00**	**0**	**0.00**	**0**	**0.00**
unix	3	**2**	0.05	**2**	**0.01**	**1**	0.04	**1**	**0.02**
world	50	**46**	—	50	—	15	—	**13**	—

To compare speed and effectiveness of the model we ran it on two sets of graphs. The first set of graphs are all the hierarchical network diagrams appearing in the GraphViz gallery [3]. The second set of graphs are random graphs of k-levels with n nodes per level and a fixed edge density of 20% (that is each node is connected on average to 20% of the nodes on the next layer); these do not include any long edges. Problem class gk_n is a suite of 10 randomly generated instances with k levels and n nodes per level.

Table 1 shows the results of minimizing edge crossings and maximizing planar subgraphs with MIP and SAT solvers, as well as the crossings resulting in the Graphviz heuristic layout for graphs from the GraphViz gallery. We use the best variation of our model for each solver: for the MIP solver this is with all improvements described in the previous section, while for the SAT solver we omit the leaf optimization since it slows down the solver. For each solver we show the solution, with least edge crossings or minimal number of edges deleted for planarity, found in 60s and the time to prove optimality or '—' if it was not proved optimal. The results show that for realistic graphs we can find better solutions than the heuristic method, even when there are very few crossings. The best found crossing and edges deleted for planarization and problems solved/time combination are highlighted in bold. We can find optimal crossing solutions for 9 out of ten examples, and maximal planar subgraph solutions for 7 out of 10 examples. The MIP approach is clearly superior for minimizing edge crossings, while SAT is superior for maximizing planarity.

Table 2 shows the results of crossing minimization and maximal planar subgraph for the second data set of random graphs using the MIP and SAT solver. The table shows: the total number of crossings when the graphs are laid out using GraphViz then for each solver: the total number of crossings or edge deletions in the best solutions found in 60s for the suite (a '—' indicates that for at least one instance the method found no solution better than the Graphviz

Table 2. Time to find and prove the minimal crossing layout and maximal planar subgraph, using MIP and SAT for random examples

Problem	Crossing minimization					Maximal planar subgraph			
	graphviz	MIP		SAT		MIP		SAT	
	best	best	solved	best	solved	best	solved	best	solved
g3_7	41	**35**	**10 / 0.00**	**35**	10 / 0.02	**18**	**10 / 0.01**	**18**	10 / 0.02
g3_8	103	**86**	**10 / 0.03**	**86**	10 / 0.10	**31**	10 / 0.15	**31**	**10 / 0.04**
g3_9	204	**195**	**10 / 0.07**	**195**	10 / 6.67	**63**	10 / 4.60	**63**	**10 / 0.12**
g3_10	399	**373**	**10 / 0.97**	380	8 / 14.44	**91**	5 / 15.53	**91**	**10 / 3.37**
g4_7	70	**55**	**10 / 0.01**	**55**	10 / 0.04	**32**	10 / 0.11	**32**	**10 / 0.04**
g4_8	187	**169**	**10 / 0.09**	**169**	10 / 0.68	**61**	10 / 3.83	**61**	**10 / 0.16**
g4_9	351	**342**	**10 / 0.89**	345	8 / 13.69	**94**	6 / 26.85	**94**	**10 / 2.82**
g4_10	703	**681**	**9 / 2.37**	—	—	161	—	**152**	—
g5_7	101	**95**	**10 / 0.03**	**95**	10 / 0.11	**47**	10 / 0.43	**47**	**10 / 0.08**
g5_8	284	**245**	**10 / 0.32**	**245**	10 / 4.01	**93**	10 / 25.46	**93**	**10 / 2.78**
g5_9	474	**450**	**10 / 0.99**	—	5 / 30.47	139	1 / 35.37	**138**	**3 / 47.46**
g6_7	141	**131**	**10 / 0.03**	**131**	10 / 0.18	**57**	10 / 1.51	**57**	**10 / 0.18**
g6_8	357	**324**	**10 / 0.53**	**324**	10 / 13.34	112	4 / 11.64	**111**	**10 / 28.81**
g6_9	684	**637**	**10 / 3.28**	—	—	**190**	—	197	—
g7_7	159	**148**	**10 / 0.08**	**148**	10 / 0.58	**67**	10 / 3.91	**67**	**10 / 0.78**
g7_8	390	**366**	**10 / 0.72**	372	6 / 12.35	**134**	**1 / 47.75**	140	—
g7_9	813	**786**	**9 / 12.54**	—	—	238	—	**233**	—
g8_7	249	**235**	**10 / 0.16**	**235**	10 / 4.06	**92**	8 / 13.38	**92**	**10 / 4.91**
g8_8	466	**431**	**10 / 1.51**	—	1 / 10.73	**154**	—	165	—
g9_7	269	**238**	**10 / 0.30**	**238**	10 / 2.60	**108**	7 / 17.72	**108**	**8 / 15.18**
g9_8	572	**541**	**10 / 2.95**	—	3 / 28.71	**197**	—	200	—
g10_7	334	**304**	**10 / 0.27**	**304**	10 / 9.67	**119**	**8 / 24.05**	121	4 / 19.39
g10_8	733	**661**	**10 / 10.68**	—	—	**216**	—	225	—

bound in 60s) and the number of instances where optimal solutions were found and proved and the average time to prove optimality.

The results are in accord with those for the first dataset and show that the MIP solver can almost always find optimal minimal crossing solutions within this time bound (only two instances failed). The Graphviz solutions can be substantially improved, the best solutions found have 10-20% fewer crossings.

For maximal planar subgraph, in contrast to edge crossings, the SAT solver is better than the MIP solver, although as the number of levels increases the advantage decreases.

Tables 3 and 4 show the results for the mixed objective functions: minimizing crossings then maximizing planar subgraph and the reverse. For minimizing crossings first MIP dominates as before, and again is able to solve almost all problems optimally within 60s. For the reverse objective SAT is better for the small instances, but suffers as the instances get larger. This problem is significantly harder than the minimizing crossings first.

Results not presented demonstrate that the improvements presented in the previous section make a substantial difference. The elimination of $K_{2,2}$ cycles is

Table 3. Time to find and prove optimal mixed objective solutions for Graphviz examples using MIP and SAT

Problem	Crossing then planarization				Planarization then crossing			
	MIP		SAT		MIP		SAT	
	best	solved	best	solved	best	solved	best	solved
crazy	**(2, 1)**	**0.07**	**(2, 1)**	10.51	**(1, 2)**	**0.08**	**(1, 2)**	0.31
datastruct	**(2, 1)**	**0.02**	**(2, 1)**	0.26	**(1, 2)**	**0.03**	**(1, 2)**	0.23
fsm	**(0, 0)**	**0.00**	**(0, 0)**	**0.00**	**(0, 0)**	**0.00**	**(0, 0)**	**0.00**
lion_share	**(4, 3)**	**0.15**	**(4, 3)**	3.55	**(2, 5)**	**0.52**	**(2, 5)**	0.60
profile	**(38, 11)**	**29.96**	(281, 34)	—	**(12, 66)**	—	(13, 145)	—
switch	**(20, 17)**	**1.36**	**(20, 17)**	3.61	**(17, 20)**	—	**(17, 20)**	—
traffic_lights	**(0, 0)**	**0.00**	**(0, 0)**	**0.00**	**(0, 0)**	**0.00**	**(0, 0)**	0.01
unix	**(2, 1)**	**0.07**	**(2, 1)**	10.52	**(1, 2)**	**0.09**	**(1, 2)**	0.32
world	**(47, 14)**	—	(108, 19)	—	(18, 79)	—	**(15, 106)**	—

Table 4. Time to find and prove optimal mixed objective solutions for random examples using MIP and SAT

Problem	Crossing then planarization				Planarization then crossing			
	MIP		SAT		MIP		SAT	
	best	solved	best	solved	best	solved	best	solved
g3_7	**(35, 22)**	**10 / 0.01**	**(35, 22)**	10 / 0.04	**(18, 39)**	**10 / 0.02**	**(18, 39)**	10 / 0.04
g3_8	**(86, 41)**	**10 / 0.13**	**(86, 41)**	10 / 0.37	**(31, 102)**	10 / 0.31	**(31, 102)**	**10 / 0.21**
g3_9	**(195, 78)**	**10 / 0.48**	**(195, 78)**	10 / 3.09	**(63, 231)**	9 / 10.31	**(63, 231)**	**10 / 1.04**
g3_10	**(373, 115)**	**10 / 2.67**	(564, 166)	2 / 40.99	(91, 444)	2 / 13.84	**(91, 419)**	**9 / 10.64**
g4_7	**(55, 35)**	**10 / 0.05**	**(55, 35)**	10 / 0.23	**(32, 58)**	**10 / 0.21**	**(32, 58)**	10 / 0.23
g4_8	**(169, 76)**	**10 / 0.57**	**(169, 76)**	10 / 1.58	**(61, 223)**	10 / 6.66	**(61, 223)**	**10 / 1.91**
g4_9	**(342, 116)**	**10 / 1.66**	(418, 162)	5 / 33.21	**(94, 386)**	3 / 31.03	(95, 382)	**8 / 14.76**
g4_10	**(681, 195)**	**9 / 8.17**	(1249, 338)	—	**(158, 933)**	—	(160, 928)	—
g5_7	**(95, 55)**	**10 / 0.08**	**(95, 55)**	10 / 0.35	**(47, 104)**	10 / 0.76	**(47, 104)**	**10 / 0.53**
g5_8	**(245, 108)**	**10 / 0.66**	**(245, 108)**	10 / 8.38	(95, 269)	4 / 29.48	**(94, 290)**	**8 / 25.24**
g5_9	**(450, 174)**	**10 / 3.83**	(694, 210)	1 / 42.43	**(142, 612)**	—	(146, 656)	—
g6_7	**(131, 64)**	**10 / 0.25**	**(131, 64)**	10 / 1.98	**(57, 153)**	10 / 2.74	**(57, 153)**	**10 / 2.32**
g6_8	**(324, 136)**	**10 / 1.16**	(357, 150)	6 / 22.50	**(112, 419)**	**2 / 22.62**	(117, 413)	2 / 31.29
g6_9	**(637, 228)**	**10 / 8.15**	(1353, 513)	—	**(192, 881)**	—	(212, 967)	—
g7_7	**(148, 83)**	**10 / 0.34**	**(148, 83)**	10 / 23.94	**(67, 168)**	10 / 10.66	**(67, 168)**	**10 / 10.59**
g7_8	**(366, 159)**	**10 / 3.00**	(454, 236)	2 / 20.16	**(136, 472)**	—	(148, 500)	—
g7_9	**(778, 255)**	**8 / 18.96**	(1372, 481)	—	**(236, 1031)**	—	(258, 1303)	—
g8_7	**(235, 116)**	**10 / 0.50**	**(235, 116)**	10 / 14.06	**(92, 272)**	5 / 15.54	(93, 277)	**8 / 22.09**
g8_8	**(431, 195)**	**10 / 5.06**	(641, 345)	1 / 33.37	**(154, 552)**	—	(182, 639)	—
g9_7	**(238, 123)**	**10 / 0.77**	(241, 126)	9 / 25.00	**(108, 260)**	**6 / 16.29**	(112, 283)	2 / 57.22
g9_8	**(541, 229)**	**10 / 6.17**	(981, 464)	—	**(198, 757)**	—	(216, 871)	—
g10_7	**(304, 144)**	**10 / 1.59**	(329, 201)	7 / 33.19	**(119, 362)**	**4 / 32.01**	(126, 415)	1 / 58.29
g10_8	**(661, 256)**	**9 / 15.15**	(1216, 546)	—	**(199, 832)**	—	(224, 987)	—

highly beneficial to both solvers. Constraints for larger cycles can have significant benefit for the MIP solver but rarely benefit the SAT solver. The leaf optimization is good for the MIP solver, but simply slows down the SAT solver. We believe this is because it complicates the MiniSAT+ translation of the objective function to clauses. Overall the optimizations improve speed by around 2-5×.

They allow 6 more instances to find optimal solutions for minimizing crossing, 5 for maximal planar subgraph, 19 for crossing minimization then maximal planar subgraph, and 9 for maximal planar subgraph then crossing minimization.

5 Conclusion

This paper demonstrates that maximizing clarity of heirarchical network diagrams by edge crossing minimization or maximal planar subgraph or their combination can be solved optimally for reasonable sized graphs using modern SAT and MIP software. Using this generic solving technology allows us to experiment with other notions of clarity combining or modifying these notions. It also gives us the ability to accurately measure the effectiveness of heuristic methods for solving these problems.

References

1. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, Englewood Cliffs (1999)
2. Mutzel, P.: An alternative method to crossing minimization on hierarchical graphs. In: Graph Drawing, pp. 318–333 (1997)
3. Gansner, E., North, S.: An open graph visualization system and its applications to software engineering. Software: Practice and Experience 30(11), 1203–1233 (2000)
4. Sugiyama, K., Tagawa, S., Toda, M.: Methods for visual understanding of hierarchical system structures. IEEE Trans. Syst. Man Cybern. 11(2), 109–125 (1981)
5. Matuszewski, C., Schönfeld, R., Molitor, P.: Using sifting for k-layer straightline crossing minimization. In: Kratochvíl, J. (ed.) GD 1999. LNCS, vol. 1731, pp. 217–224. Springer, Heidelberg (1999)
6. Jünger, M., Mutzel, P.: 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. Journal of Graph Algorithms and Applications 1(1), 1–25 (1997)
7. Jünger, M., Lee, E.K., Mutzel, P., Odenthal, T.: A polyhedral approach to the multi-layer crossing minimization problem. In: DiBattista, G. (ed.) GD 1997. LNCS, vol. 1353, pp. 13–24. Springer, Heidelberg (1997)
8. Randerath, B., Speckenmeyer, E., Boros, E., Hammer, P., Kogan, A., Makino, K., Simeone, B., Cepek, O.: A satisfiability formulation of problems on level graphs. Electronic Notes in Discrete Mathematics 9, 269–277 (2001)
9. Healy, P., Kuusik, A.: The vertex-exchange graph: A new concept for multi-level crossing minimisation. In: Kratochvíl, J. (ed.) GD 1999. LNCS, vol. 1731, pp. 205–216. Springer, Heidelberg (1999)
10. Eén, N., Sörensson, N.: Translating pseudo-boolean constraints into SAT. Journal on Satisfiability, Boolean Modeling and Computation 2, 1–26 (2006)