Using Sifting for k-Layer Straightline Crossing Minimization

Christian Matuszewski Robby Schönfeld Paul Molitor
Institute for Computer Science
University Halle-Wittenberg, Germany

June 24, 1999

Abstract

We present a new algorithm for k-layer straightline crossing minimization which is based on sifting that is a heuristic for dynamic reordering of decision diagrams used during logic synthesis and formal verification of logic circuits. The experiments proves the approach to be very efficient. Especially, it outperforms the heuristics known from literature by far when applied to k-layered graphs with $k \geq 3$.

1 Introduction

Directed graphs are commonly used to represent information in many fields such as software engineering, project management, and social sciences. A good visualization of this information is necessary to get general ideas of relations.

A common method for drawing acyclic graphs was introduced by [STT81] and [Car80]. This approach consists of four phases.

1. In a first step, the directed graph is made acyclic by (temporarily) reversing some edges. Let $G = (V, E)$ be the resulting directed acyclic graph.
2. Then, the set V of the nodes is partitioned into k layers $V_1, \ldots, V_k$ $(k \geq 1)$ such that for any edge $(u, v) \in E$ with $u \in V_i$ and $v \in V_j$, the inequation $i < j$ holds. Subset V_i is called i^{th} layer. G is called a k-layered directed graph.

   Usually, a k-layer directed graph is drawn in such a way that the vertices of each layer V_i lie on a horizontal line. All the edges are drawn as straight lines.
3. In the third step, the vertices within each layer are permuted targeting minimization of the number of crossings.
4. In the last step, the vertices and the edges are placed according to the permutations computed in step 3. Note that the number of crossings does not depend on the exact placement, but only on the permutations of the vertices in the layers.

In this paper, we focus on step 3, namely permuting the vertices within the layers in such a way that the number of crossing is minimized. We assume that graph $G = (V_1 \cup \ldots \cup V_k, E)$ has been made *proper* in step 2, i.e., that for any edge (u, v) with $u \in V_i$ and $v \in V_j$ the equation $j - i = 1$ holds. This can be achieved by inserting dummy vertices on edges connecting vertices on non-neighboring layers.

As already mentioned, the number of crossings only depends on the ordering of the vertices in the layers and not on the exact positions of the vertices because edges are drawn as straight lines. Thus, the problem is to find a permutation π_i for every layer V_i such that the number of edge crossings is minimized. Unfortunately, the problem of finding a permutation π_i for every layer V_i such that the number of crossings is minimal is NP-hard even for 2-layered directed graphs [GJ83]. Furthermore, the problem of 2-layer straightline crossing minimization remains NP-hard when the permutation of one layer is fixed [EW94a]. In the following, we refer to the problem of crossing minimization in 2-layered graphs with both layers freely permutable as the *two sided crossing minimization problem.* If the permutation of one layer is fixed, we call the problem the *one sided crossing minimization problem.*

The paper is structured as follows. In Section 2 we shortly review heuristics known from literature. Section 3 presents the new approach which is a standard technique for minimization of binary decision diagrams (BDDs) during logic synthesis and formal verification of logic circuits. The experimental procedure of this section only concentrates on the one sided crossing minimization problem. We generalize sifting to general k-layer straightline crossing minimization in Section 4. Experimental results prove the new approach to be very efficient, especially for the general problem and $k \geq 3$. Mostly, it outperforms the heuristics known from literature by far.

2 Algorithms known from literature

Most methods to reduce the crossings in a k-layered directed graph use a technique called the *layer-by-layer sweep* which works as follows. First, an initial vertex ordering for every layer is computed. Then, for $i = 2, \ldots, k$, the vertex ordering of layer V_{i-1} is held fixed while the vertices of layer V_i are reordered to reduce crossings. After that, the method sweeps back holding fixed the permutation of layer V_i and permute the vertices in layer V_{i-1}. This process is repeated until no further refinement can be achieved. With this approach the k-layer crossing minimization problem is reduced to a series of one sided crossing minimization problems.

Let us consider the one sided crossing minimization problem in more detail. Without loss of generality, let the permutation π_1 of layer V_1 be fixed. For every pair of vertices $u, v \in V_2$ we define the *crossing number* c_{uv} as the number of edge crossings that edges incident to u make with edges incident to v if $\pi_2(u) < \pi_2(v)$ holds. Thus,

$$\mathrm{cross}(G, \pi_1, \pi_2) = \sum_{\pi_2(u) < \pi_2(v)} c_{uv} \tag{1}$$

is the number of crossings in the straightline drawing of the 2-layered graph $G = (V_1 \cup V_2, E)$ with respect to the permutations π_1 and π_2. An obvious lower bound of $\mathrm{cross}(G, \pi_1, \pi_2)$ is given by

$$L = \sum_{\pi_2(u) < \pi_2(v)} \min\{c_{uv}, c_{vu}\}.$$

Experiments in [JM97] have proven this lower bound to be very tight to the optimum.

The most popular heuristics for one sided crossing minimization are the barycenter heuristic [STT81] and the median heuristic [EW94b]. They simply compute the barycenters or the medians of the adjacent nodes of each node v in the neighboring layer and sort the vertices according to these numbers. Another heuristic applied to one sided crossing minimization is the split heuristic [EK86] that chooses a pivot vertex p and places every other vertex to the left or right of this pivot vertex according to whether it would make fewer crossings. This step is applied recursively to the set left of p and to the set right of p. The greedy-insert heuristic [EK86] proceeds by successively choosing the next vertex to be the one which minimizes the number of crossings that edges adjacent to it make with edges adjacent to vertices to the left of it. The greedy-switch heuristic [EK86] scans all consecutive pairs of vertices and switches them if this reduces the number of crossings. This process is repeated until no switching occurs any more. Finally, Jünger and Mutzel [JM97] presented a very efficient branch and cut algorithm for one sided crossing minimization practically applicable to graphs up to 60 vertices. In their paper, they also experimentally analyzed the quality of the heuristics when applied to one sided crossing minimization. They proved that the barycenter heuristic and the split heuristic perform very well in terms of quality. Greedy-insert behaves worst.

For two sided crossing minimization, Jünger and Mutzel [JM97] have presented an exact algorithm that can compute true optima for instances in which the smaller layer contains (only) up to 15 nodes. They experimentally proved that for sparse instances, the optimal solution is much better than any of the solutions found by the heuristics considered. The iterated barycenter method turns out to be the best method compared to all other iterated heuristics.

3 Sifting for One Sided Crossing Minimization

In this section, we present a new heuristic for straightline crossing minimization, called *Sifting*. Sifting was first introduced by Rudell [Rud93] to reduce the number of vertices in reduced ordered binary decision diagrams (ROBDDs). The algorithm can be easily adapted to the one sided crossing minimization problem.

Assume that permutation π_1 of layer V_1 is fixed. We choose a vertex v from layer V_2 to put it on a position which minimizes the number of crossings, whereby the ordering of all other vertices in layer V_2 remains fixed. In order to do so, vertex v is moved to the leftmost position by repeatedly swapping it with its left neighbors. Then, it is moved to the right. After reaching the rightmost position vertex, v is moved to its (locally) optimal location. Then, the next vertex will be sifted.

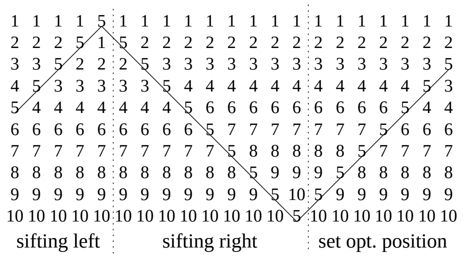

Figure 1: Sifting of the vertex labeled by 5.

To find the best position for a given vertex, we have to compute the number of crossings after each swapping. This can easily be done. Assume we have to swap the two vertices u and v with u being the left neighbor of v. Then, after the swapping step there are

$$cross_{after\_swapping} = cross_{before\_swapping} - c_{uv} + c_{vu}$$

crossings. With that, sifting of one vertex takes $\mathcal{O}(n)$ resulting in an overall runtime of $\mathcal{O}(n^2)$, where n is the number of vertices.

There are different possibilities for choosing the next vertex to be sifted. One method is to use a preassigned vertex order of the layer, e.g., from left to right (method 1). A more accurate approach seems to be to sift the vertices according to their degrees, i.e., sift the vertices with high indegree, first (method 2). The third method investigated by us randomly chooses the vertices.

Experimental results for one sided crossing minimization

Our computational experiments include the *barycenter* heuristic, the *median* heuristic, the *greedy-insert* heuristic, the *greedy-switch* heuristic, the *split* heuristic, and the *sifting* heuristics (method 1 up to method 3). For our experiments we used the algorithms implemented in the AGD library [MGB+98] which is based on LEDA [MN99].

We used the program `random_bigraph` of the Stanford GraphBase [Knu93] with the same parameters as in the paper of Jünger and Mutzel [JM97]. We could reproduce the results of Jünger and Mutzel exactly except for the median heuristic and the barycenter heuristic. The differences with the median heuristic are due to the fact that the implementation in AGD uses the *average median* from [Mäk90] rather than the original median heuristic from [EW94b]. Our results are always better than those given in [JM97]. The slight deviation from the results with the barycenter heuristic are possibly due to minor differences in sorting methods.

In the first experimental run for one sided crossing minimization, we considered sparse graphs with an increasing number of vertices, 10 samples for each type of graphs. Such instances are among the most interesting in practical applications. In our experiments we use sparse graphs with $|E| = |V_1| + |V_2|$. That means each vertex has on the average 2 edges incident to it. Figure 3 gives the relative size of the average number of crossings taken over all sampled instances of the given graph type in percentage of the minimum number of crossings, which has been computed by the branch and cut algorithm for one sided crossing minimization of Jünger and Mutzel [JM97]. The curves prove that all three sifting methods are close to the optimum for sparse graphs and that sifting dominates the heuristics known from literature for that class of graphs. The running time of the sifting algorithm is about 1 second for 100+100-graphs, i.e., graphs $G = (V_1 \cup V_2, E)$ with $V_1 = V_2 = 100$.

The next experimental run we have made considers 20+20-graphs with increasing edge densities from 10 till 90 percent. Figure 4 gives the result for 100 samples for each type of graph. It shows that for one sided crossing minimization sifting dominates the other heuristics for small densities. In the case of high densities, the barycenter heuristic and the split heuristic are as efficient as sifting is.

The experiments presented show that the vertex order used during sifting does not influence the quality of the heuristic very much. Furthermore, the experiments show that sifting applied to one sided crossing minimization is a very efficient heuristic.

4 Extending Sifting to k-layered directed graphs

As stated in [JM97], there is no real need for heuristics for one sided crossing minimization up to 60 vertices in the permutable layer as the optimum solution can be computed fast by the branch and cut algorithm. So, let us concentrate on the general problem, namely on the k-layer straightline crossing minimization problem for $k \geq 2$, in the following.

Sifting can obviously be extended to k-layer straightline crossing minimization by using sifting based layer-by-layer sweep. Another method is to keep a list $l = (l_1, \ldots, l_n)$ of all vertices $l_i \in V$, $n = |V|$ from graph $G = (V, E) = (V_1 \cup \ldots \cup V_k, E)$ in descending order of the degree of vertex l_i which is the number of incident edges to vertex l_i and to sift each l_i in its associated layer according to this order. If no improvement is made, we reverse the vertex order of list l and memorize it as a fail, otherwise we use the same sequence for the next trial again. After these two loop passes, the vertex order is inverted in any case and the cycle is repeated if the number of fails is less than a defined maximum count. Despite the possibility of better results, we have not taken care about this constant and set it to zero in all experimental runs. Note that in this algorithm optimization is not only realized layer-by-layer. Figure 2 presents the algorithm which we denote by `Global Sifting` in detail.

```
failcount = 0
do
    forall(l,l_i) sift_node(l_i)
    if(fail)
        reverse(l)
        failcount++

    forall(l,l_i) sift_node(l_i)
    if(fail) failcount++

    reverse(l)

until(failcount ≥ MAXFAILS)
```

Figure 2: Non-fixed layer algorithm (Global Sifting).

Experimental results for k-layered graphs

In our experiments, we have considered directed graphs up to 12 layers. All heuristics, except global sifting, are iterated between the k layers until a local optimum is obtained.

2-layered directed graphs

For crossing minimization of small sparse 2-layered graphs $G = (V_1 \cup V_2, E)$ with $|E| = |V_1| + |V_2|$ sifting is slightly better than the barycenter method, whereas barycenter dominates sifting for larger sparse 2-layered graphs. Figure 5 illustrates the results obtained. The horizontal curve at the 100 percentage level represents the crossing minimization results of the barycenter heuristic.

The insufficient quality of sifting based layer-by-layer sweep and global sifting for increasing number of vertices seems to be due to the fact that the density decreases when the number of nodes increases. A sparse 10+10-graph, e.g., contains 20 edges resulting in a density of 20 percent, whereas a 100+100-graph which contains 200 edges only have a density of 2 percent. This is the reason we have repeated our experimental run for increasing number of nodes with a constant edge density of 20 percent. Figure 6 gives evidence of our assumption. It shows that sifting and global sifting are slightly better than the barycenter method when applied to 2-layered graphs with edge density greater than 15 percent.

4-layered directed graphs

As the order in which the vertices are sifted during one-sided crossing minimization does not influence the quality of the results very much, we only consider layer-by-layer sweep based on method 1 of sifting and global sifting, in the following. We compare them to the barycenter, the median, the split, the greedy-insert and the greedy-switch heuristic. We added the last two heuristics because they have not been probed on k-layered directed graphs for $k \geq 3$, yet.

For more than 2 layers, global sifting dominates all the other heuristics by far. This fact is illustrated by Figure 7, 8 and 9. Here, the horizontal curve at the 100 percentage level represents the results for global sifting.

In our experiments, we used k*b-graphs, i.e., k-layered directed graphs $G_k = (V_1 \cup \ldots \cup V_k, E)$ with $|V_i| = b$ for all $1 \leq i \leq k$.

In a first experimental run, we have varied the number of vertices in sparse graphs with 4 layers. For this, we extended the concept of sparse graphs from 2 layers to k layer by only considering k*b-graphs $G = (V_1 \cup \ldots \cup V_k, E)$ with $|E| = 2 \cdot (k-1) \cdot b$. On the average, a vertice should have 2 neighbors in each adjacent layer. Figure 7 shows the results for sparse 4*10-graphs up to 4*50-graphs. Surprisingly, sifting based layer-by-layer sweep is much worse than the barycenter heuristic, the median heuristic, and the split heuristic. However, global sifting is very efficient and outperforms all the other methods.

The next experimental run, we have made, considers k*b-graphs with higher densities. The results obtained are shown in Figure 8. Although all the heuristics converge to the true optima when the edge density is increased, global sifting is much more efficient for dense graphs than all the other heuristics, too.

Finally, we studied the influence of the number of layers on the quality of the crossing minimization (see Figure 9). For this, we have used sparse graphs with 10 vertices per layer. The number of layers is rising from 2 till 12 layers. It can be observed that the layer-by-layer sweep based on any heuristic lose performance against global sifting for increasing number of layers.

5 Conclusions

We have presented a new approach for k-layer straightline crossing minimization which has been proven to be very efficient. The experiments lead to the conclusion that global sifting dominates the heuristics known from literature, in most cases, especially for k-layer straightline minimization for $g \geq 3$.

Acknowledgements

We would like to thank Petra Mutzel, Thomas Ziegler, and Stefan Näher for helpful discussions concerning the AGD library and LEDA.

References

[Car80] M. J. Carpano. Automatic display of hierarchized graphs for computer aided decision analysis. *IEEE Trans. Syst. Man Cybern.*, SMC-10(11):705–715, 1980.

[EK86] P. Eades and D. Kelly. Heuristics for reducing crossings in 2-layered networks. *Ars Combin.*, 21.A:89–98, 1986.

[EW94a] P. Eades and S. Whitesides. Drawing graphs in two layers. *Theoretical Computer Science*, 131:361–374, 1994.

[EW94b] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994.

[GJ83] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4:312–316, 1983.

[JM97] M. Jünger and P. Mutzel. 2-Layer straightline crossing minimization: Performance of exact and heuristic algorithms. *J. Graph Algorithms Appl.*, 1(1):1–25, 1997.

[Knu93] D. E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Comput ing*. Addison-Wesley, Reading, MA, 1993.

[Mäk90] E. Mäkinen. Experiments on drawing 2-level hierarchical graphs. *Internat. J. Comput. Math.*, 36:175–181, 1990.

[MGB+98] P. Mutzel, C. Gutwenger, R. Brockenauer, S. Fialko, G. W. Klau, M. Krüger, T. Ziegler, S. Näher, D. Alberts, D. Ambras, G. Koch, M. Jünger, C. Buchheim, and S. Leipert. A library of algorithms for graph drawing. In S. H. Whitesides, editor, *Proceedings of the 6th International Symposium on Graph Drawing (GD '98)*, volume 1547 of *Lecture Notes in Computer Science*, pages 456–457. Springer, 1998. Project home page at `http://www.mpi-sb.mpg.de/AGD/`.

[MN99] K. Mehlhorn and S. Näher. *The* LEDA *Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999. Project home page at `http://www.mpi-sb.mpg.de/LEDA/`.

[Rud93] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Proc. International Conf. on Computer-Aided Design (ICCAD)*, pages 42–47, November 1993.

[STT81] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, February 1981.

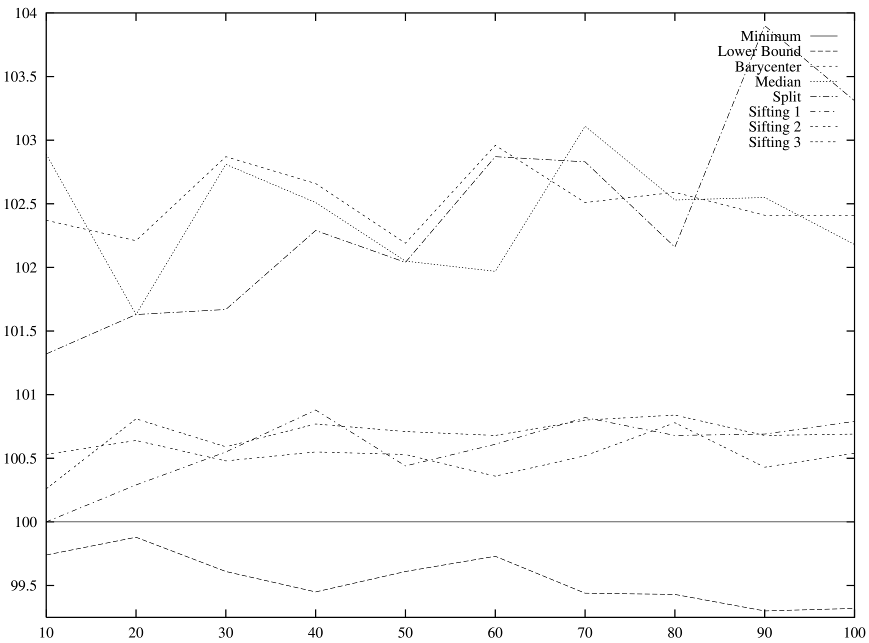

Figure 3: Results for one sided crossing minimization, sparse graphs, increasing number of vertices, 10 samples.

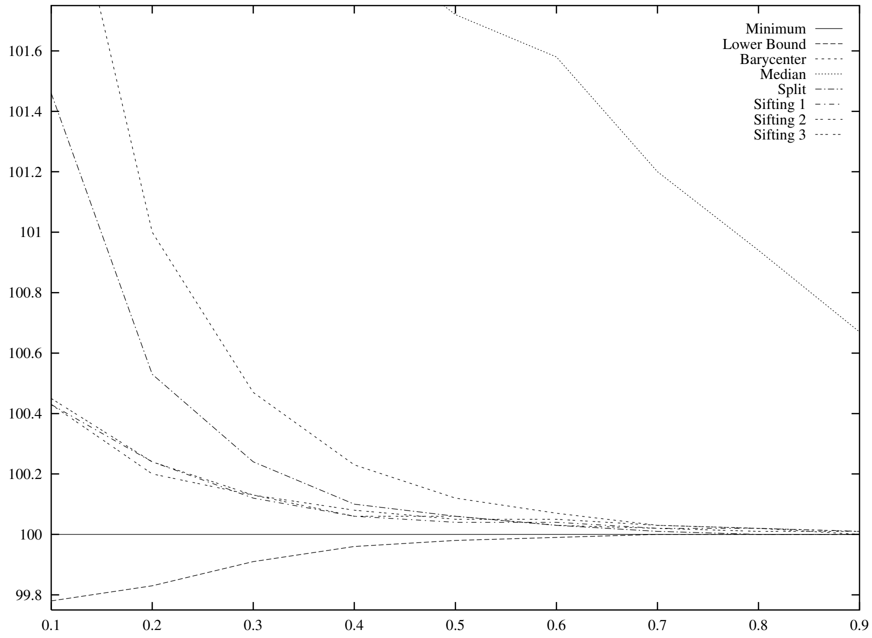

Figure 4: Results for one sided crossing minimization, 20+20-graphs, increasing edge density, 100 samples.

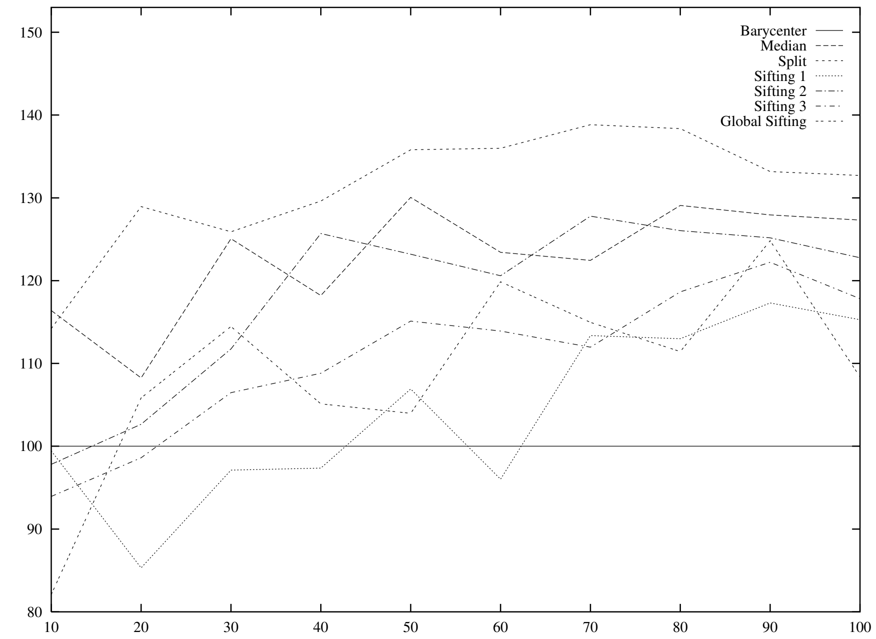

Figure 5: Results for two sided crossing minimization, sparse graphs, increasing number of vertices, 10 samples.

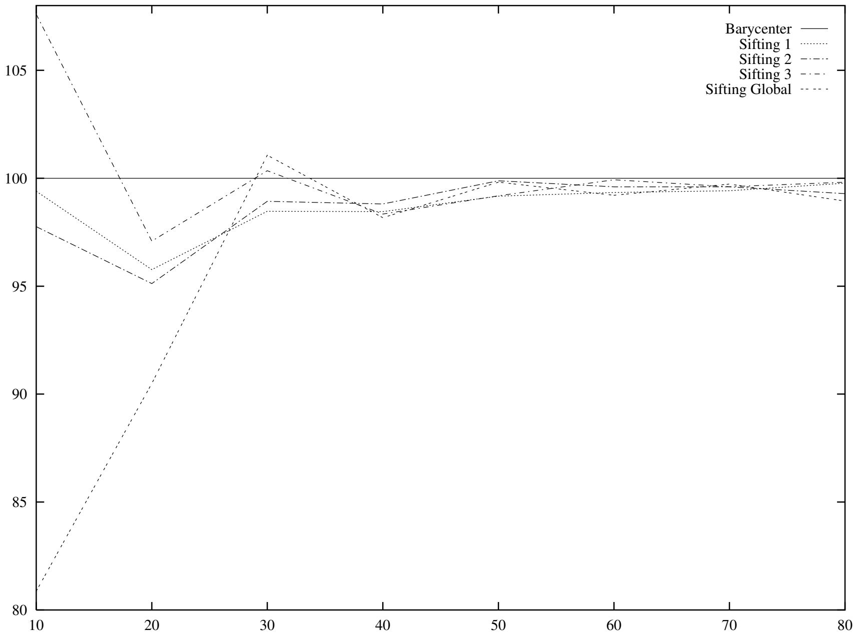

Figure 6: Results for two sided crossing minimization, graphs with constant density 0.2, increasing number of vertices, 10 samples.

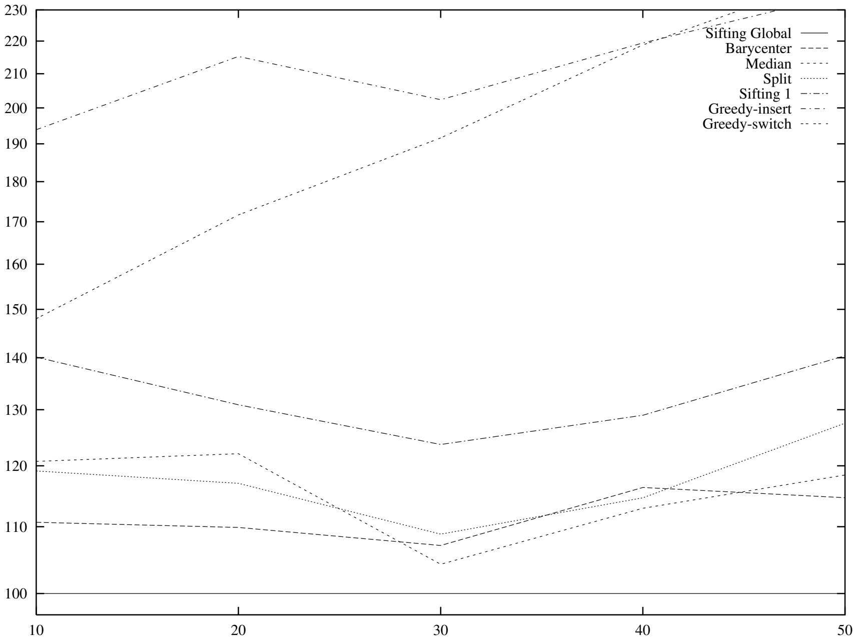

Figure 7: Results for 4-layer crossing minimization, sparse graphs, increasing number of vertices, 10 samples.

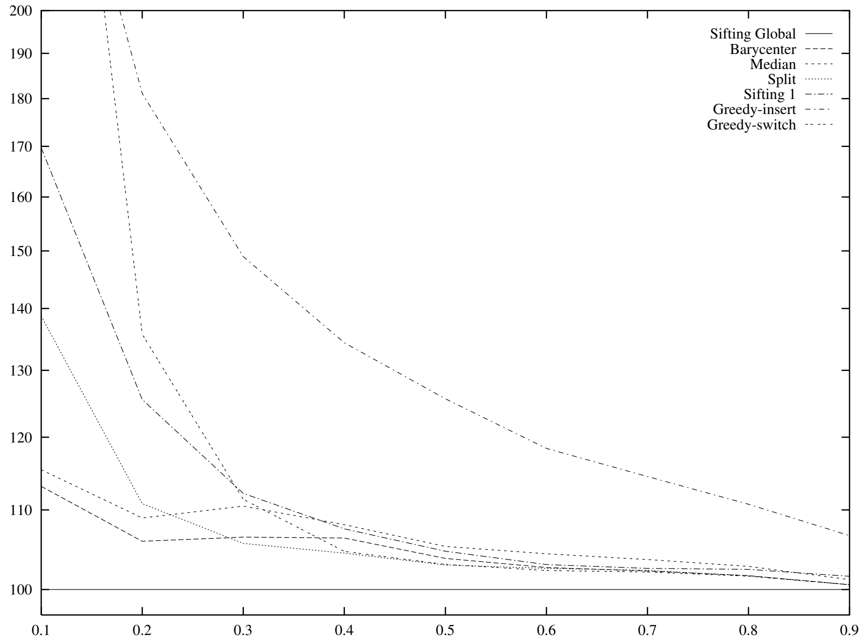

Figure 8: Results for 4-layer crossing minimization, 4*10-graphs, increasing edge density, 100 samples.

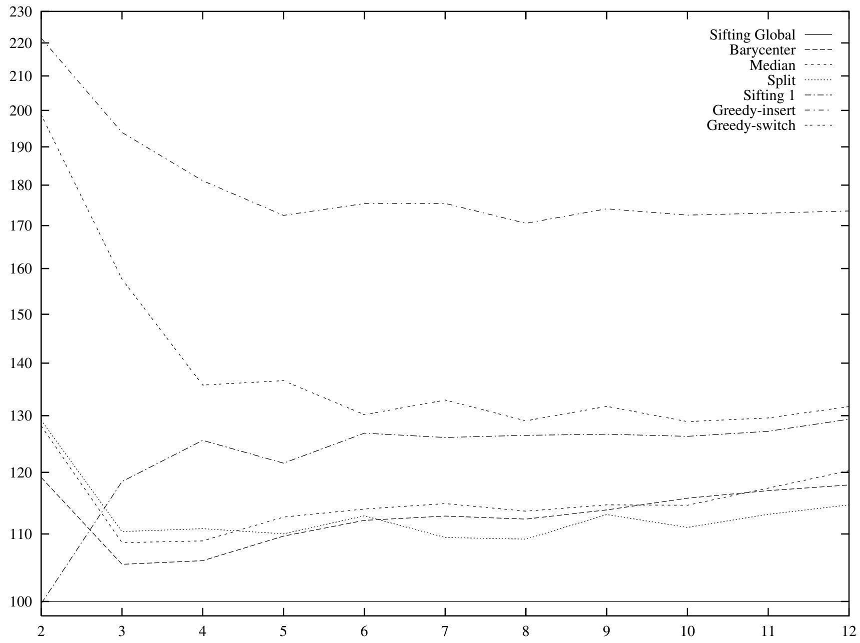

Figure 9: Results for k-layer crossing minimization, sparse graphs, 10 vertices per layer, increasing number of layers, 100 samples.