Journal of Graph Algorithms and Applications

http://www.cs.brown.edu/publications/jgaa/

vol. 1, no. 1, pp. 1–25 (1997)

2-Layer Straightline Crossing Minimization: Performance of Exact and Heuristic Algorithms

Michael Jünger

Institut für Informatik
Universität zu Köln
http://www.informatik.uni-koeln.de/ls_juenger/
mjuenger@informatik.uni-koeln.de

Petra Mutzel

Max-Planck-Institut für Informatik
Saarbrücken
http://www.mpi-sb.mpg/~mutzel/
mutzel@mpi-sb.mpg.de

Abstract

We present algorithms for the two layer straightline crossing minimization problem that are able to compute exact optima. Our computational results lead us to the conclusion that there is no need for heuristics if one layer is fixed, even though the problem is NP-hard, and that for the general problem with two variable layers, true optima can be computed for sparse instances in which the smaller layer contains up to 15 nodes. For bigger instances, the iterated barycenter method turns out to be the method of choice among several popular heuristics whose performance we could assess by comparing their results to optimum solutions.

Communicated by P. Eades: submitted August 1996; revised November 1996.

Research supported in part by DFG-Grant Ju204/7-1, Forschungsschwerpunkt "Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen" and by ESPRIT LTR Project no. 20244 – ALCOM-IT

1 Introduction

Directed graphs are widely used to represent structures in many fields such as economics, social sciences, mathematics and computer science. A good visualization of structural information allows the reader to focus on the information content of the diagram. Examples are entity-relationship diagrams, PERT-diagrams, or any flow diagram.

A common method for drawing directed graphs has been introduced by Sugiyama et al. [14] and Carpano [1]. In the first step, the vertices are partitioned into a set of k levels, and in the second step, the vertices within each level are permuted in such a way that the number of crossings is small. In this paper we focus on the second step.

Let us assume that we are given a *k-layered network*, i.e., a graph $G = (V, E) = (V_1, V_2, \ldots, V_k, E)$ with vertex sets $V_1, \ldots, V_k$, $V = V_1 \cup V_2 \ldots \cup V_k$, $V_i \cap V_j = \emptyset$ for $i \neq j$, and an edge set E connecting vertices in levels V_i and V_j with $i \neq j$ $(1 \leq i, j \leq k)$. V_i is called the *i-th layer.* A k-layered network is drawn in such a way that the vertices in each layer V_i are drawn on a horizontal line L_i with y-coordinate $k - i$, and the edges are drawn as straight lines. Essentially, a k-layered network is a k-partite graph that is drawn in a special way.

Even for 2-layered graphs the straightline crossing minimization problem is NP-hard [9]. The problem consists of aligning the two shores V_1 and V_2 of a bipartite graph $G = (V_1, V_2, E)$ on two parallel straight lines (layers) such that the number of crossings between the edges in E is minimized when the edges are drawn as straight lines.

Let $n_1 = |V_1|$, $n_2 = |V_2|$, $m = |E|$, and let $N(v) = \{w \in V \mid e = \{v, w\} \in E\}$ denote the set of neighbors of $v \in V = V_1 \cup V_2$ in G. Any solution is obviously completely specified by a permutation π_1 of V_1 and a permutation π_2 of V_2. For $k = 1, 2$ let $\delta^k_{ij} = 1$ if $\pi_k(i) < \pi_k(j)$ and 0 otherwise. Thus π_k $(k = 1, 2)$ is uniquely characterized by the vector $\delta^k \in \{0,1\}^{\binom{n_k}{2}}$. Given π_1 and π_2, the number of crossings is

$$\begin{aligned} C(\pi_1, \pi_2) = C(\delta^1, \delta^2) &= \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} \sum_{k \in N(i)} \sum_{l \in N(j)} \delta^1_{kl} \cdot \delta^2_{ji} + \delta^1_{lk} \cdot \delta^2_{ij} && (1) \\ &= \sum_{k=1}^{n_1-1} \sum_{l=k+1}^{n_1} \sum_{i \in N(k)} \sum_{j \in N(l)} \delta^1_{kl} \cdot \delta^2_{ji} + \delta^1_{lk} \cdot \delta^2_{ij}. && (2) \end{aligned}$$

In practice, the crossing minimization problem for k-layered networks is reduced to a series of 2-layer straightline crossing minimization problems in the following way. In a preprocessing step, we add artificial vertices to the layers L_i for all the edges traversing L_i $(i = 2, \ldots, k-1)$. For $i = 1, 2, \ldots, k-1$, we solve the 2-layer crossing minimization problem for the two adjacent layers L_i and L_{i+1} with L_i fixed, repermuting the vertices on layer L_{i+1}. Then we go backward, fixing layer L_i and repermuting the vertices on layer L_{i-1} for $i = k, k-1, \ldots, 2$. The heuristic consists of repeating these two loops until no more improvement is obtained.

Unfortunately, the 2-layer straightline crossing minimization problem with the permutation of one layer fixed is also NP-hard [7]. Therefore, a lot of effort went into the design of efficient heuristics, for the version in which one permutation is fixed as well as for the general case (see, e.g., [16, 14, 6, 8, 4, 2] and [15]). Eades and Kelly [6] observe that the computation of true optima would be desirable in order to assess the performance of various heuristics, however, they believe that the NP-hardness of the problem renders such an experimental evaluation impractical.

In this paper, we would like to demonstrate that, if one permutation is fixed, it is indeed possible to compute the exact minima in surprisingly short computation times. In section 2, we outline our algorithm which transforms the problem to a linear ordering problem that is subsequently solved via the branch and cut method. In section 3, we give computational results that allow us to assess the performance of several popular heuristics accurately.

Assume the permutation π_1 of V_1 is fixed. For each pair of nodes $i, j \in V_2$, $i \neq j$, we define c_{ij} to be the number of crossings between edges incident with i and edges incident with j if π_2 is such that $\pi_2(i) < \pi_2(j)$. Then

$$L = \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} \min\{c_{ij}, c_{ji}\}$$

is a trivial lower bound on the number of crossings. One observation in our experiments was that this trivial lower bound is surprisingly good. In section 4, we utilized this fact and the branch and cut algorithm of section 2 for the design and implementation of a program that solves the general two layer straightline crossing minimization problem to optimality.

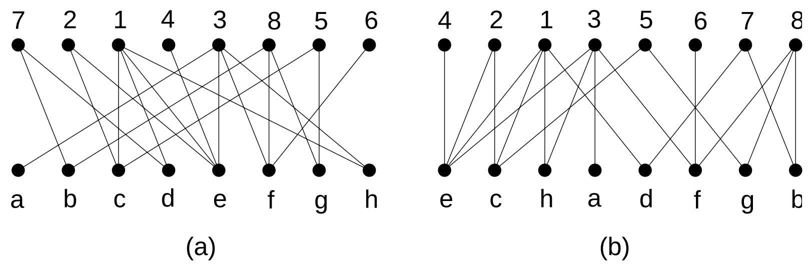

Figure 1: Crossing minimal drawings with (a) fixed lower layer and (b) both layers free

Figure 1 demonstrates that the number of crossings can indeed be considerably less if both layers can be freely permuted. The left drawing was given in [14] with fixed lower layer, [14] obtained the shown drawing with 48 crossings that we could show to be optimum. The right drawing is the optimum when both layers can be freely permuted. It has only 19 crossings.

As was to be expected, two sided crossing minimization can be done only for small instances. For large instances, we adopt the common method that consists of fixing the first layer, "optimizing" the second, fixing the found permutation of the second, "optimizing" the first, etc., back and forth, until the crossing number is not reduced anymore. We follow this iterative approach both using the heuristics of section 3 as well as the exact algorithm. The results are somewhat surprising, e.g., using the barycenter heuristic rather than exact one-sided crossing minimization yields slightly better results.

2 Branch and Cut for One Sided Crossing Minimization

The one sided straightline crossing minimization problem consists of fixing a permutation π_1 of V_1 and finding a permutation π_2 of V_2 such that the number of straightline crossings

$$C(\pi_2) = C(\delta^2) = \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} \sum_{k\in N(i)} \sum_{l\in N(j)} \delta^1_{kl} \cdot \delta^2_{ji} + \delta^1_{lk} \cdot \delta^2_{ij}$$

is minimized. Let

$$c_{ij} = \sum_{k\in N(i)} \sum_{l\in N(j)} \delta^1_{lk}$$

denote the number of crossings among the edges adjacent to i and j if $\pi_2(i) < \pi_2(j)$. Then

$$\begin{aligned} C(\pi_2) = C(\delta^2) &= \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} c_{ij}\delta^2_{ij} + c_{ji}(1-\delta^2_{ij}) && (3) \\ &= \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} (c_{ij}-c_{ji})\delta^2_{ij} + \sum_{i=1}^{n_2-1} \sum_{j=i+1}^{n_2} c_{ji}. && (4) \end{aligned}$$

For $n = n_2$, $x_{ij} = \delta^2_{ij}$ and $a_{ij} = c_{ij} - c_{ji}$ we solve the linear ordering problem

$$\begin{array}{llll} \text{(LO)} \quad \text{minimize} & \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a_{ij}x_{ij} & & (5) \\ & 0 \le x_{ij} + x_{jk} - x_{ik} \le 1 & \text{for } 1 \le i < j < k \le n & (6) \\ & 0 \le x_{ij} \le 1 & \text{for } 1 \le i < j \le n & (7) \\ & x_{ij} \text{ integral} & \text{for } 1 \le i < j \le n. & (8) \end{array}$$

If z is the optimum value of (LO), $z + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ji}$ is the minimum number of crossings.

The constraints of (LO) guarantee that the solutions correspond indeed precisely to all permutations π_2 of V_2. Furthermore, it can be shown that the "3-cycle constraints" are necessary in any minimal description of the feasible solutions by linear inequalities, if the integrality conditions are dropped. The NP-hardness of the problem makes it unlikely that such a complete linear description can be found and exploited algorithmically. Further classes of inequalities with a number of members exponential in n that must be present in a complete linear description of the feasible set, are known, and some of them can be exploited algorithmically. For the details see [12].

When the integrality conditions in (LO) are dropped, only $2\binom{n}{2}$ hypercube inequalities and $2\binom{n}{3}$ 3-cycle inequalities are left that define a relaxation of (LO) which has been proven very useful in practical applications. In [10] a branch and cut algorithm for (LO) is proposed that solves this relaxation with a cutting plane approach, since writing down all 3-cycle inequalities, even though taking only polynomial space, and solving the corresponding linear program, is not practical for space reasons. Rather, the algorithm starts with the hypercube constraints that are handled implicitly by the LP-solver, and iteratively adds violated 3-cycle constraints and deletes nonbinding 3-cycle constraints after an LP has been solved, until the relaxation is solved. If the optimum solution is integral, the algorithm stops, otherwise it is applied recursively to two subproblems in one of which a fractional x_{ij} is set to 1 and in the other set to 0. In [11] such a branch and cut approach could be used to find optimum linear orderings with n up to 60 in an application involving input-output matrices that are used in economic analysis. For the many details and the inclusion of further useful inequalities in the cutting plane part, see [10].

A new implementation of the algorithm is used in our computational experiments. It is written in C and uses the CPLEX [3] software for solving the linear programming relaxations coming up in the course of the computation.

3 One Sided Crossing Minimization

The fact that we are able to compute optimum solutions allows us to assess the quality of various popular heuristics for one-sided two layer straightline crossing minimization experimentally. Our computational comparison includes the following heuristics: the barycenter heuristic by [14], the median heuristic by [8], the stochastic heuristic by [4], the greedy-insert heuristic by [6], the greedy-switch heuristic by [6], the split heuristic by [6], and the assign heuristic by [2].

The barycenter heuristic [14] and the median heuristic [8] are the most popular ones. They are also called "averaging heuristics", since they simply compute the "average position", i.e., the barycenter or median, for each vertex and sort the vertices according to these numbers. Surprisingly, these simple heuristics turned out to be among the most promising ones. The stochastic heuristic [4], originally designed for permuting both layers, generates a series of "assessment number matrices" while greedily placing a vertex in layer 1 or layer 2. The assessment numbers are based on some frequency numbers arising from stochastic considerations on the complete bipartite graph. The greedy-insert heuristic [6] proceeds by successively choosing the next vertex v to be the one which minimizes the number of crossings that edges adjacent to v make with edges adjacent to vertices to the left of v. The greedy-switch heuristic [6] passes over all consecutive pairs of vertices and switches them if it would decrease the number of crossings. This is done until no more switching takes place. The split heuristic chooses a pivot vertex v, and places every other vertex to the left or right of v according to whether it would make fewer crossings. This step is applied recursively to order the left hand set and the right hand side of v. The assignment heuristic [2] reduces the problem to an assignment problem. The entries in the assignment matrix are computed based on the adjacency matrix and on a four dimensional matrix representing the complete bipartite graph.

In order to gain confidence in the correctness of our implementations, we repeated the computational tests in [6]. We could reproduce their results accurately. Also the results in [2] on the assign heuristic are in line with ours. There are no published computational results for the stochastic heuristic, but a personal communication with the author [5] confirms the correctness of our implementation.

All subsequent figures and tables use the following notation:

– n_i: Number of nodes on layer i for $i = 1, 2$

– m: Number of edges

– Lowerbound: The trivial lower bound for the number of crossings

– Minimum: The minimum number of crossings (computed by the branch and cut algorithm)

– Barycenter: The number of crossings found by the barycenter heuristic

– Median: The number of crossings found by the median heuristic

– Stoch: The number of crossings found by the stochastic heuristic

– Gre-ins: The number of crossings found by the greedy-insert heuristic

– Gre-swi: The number of crossings found by the greedy-switch heuristic

– Split: The number of crossings found by the split heuristic

– Assign: The number of crossings found by the assign heuristic

For each type of graph, we measured the following three numbers: the average number of crossings taken over all sampled instances of this type, the relative size of this number in percentage of the minimum number of crossings, and the average running time in seconds on a SUN Sparcstation 10. All samples are generated by the program `random_bigraph` of the Stanford GraphBase by Knuth [13]. The generators are hardware independent and are available from the authors so that exactly the same experiments can be run by anyone who is interested.

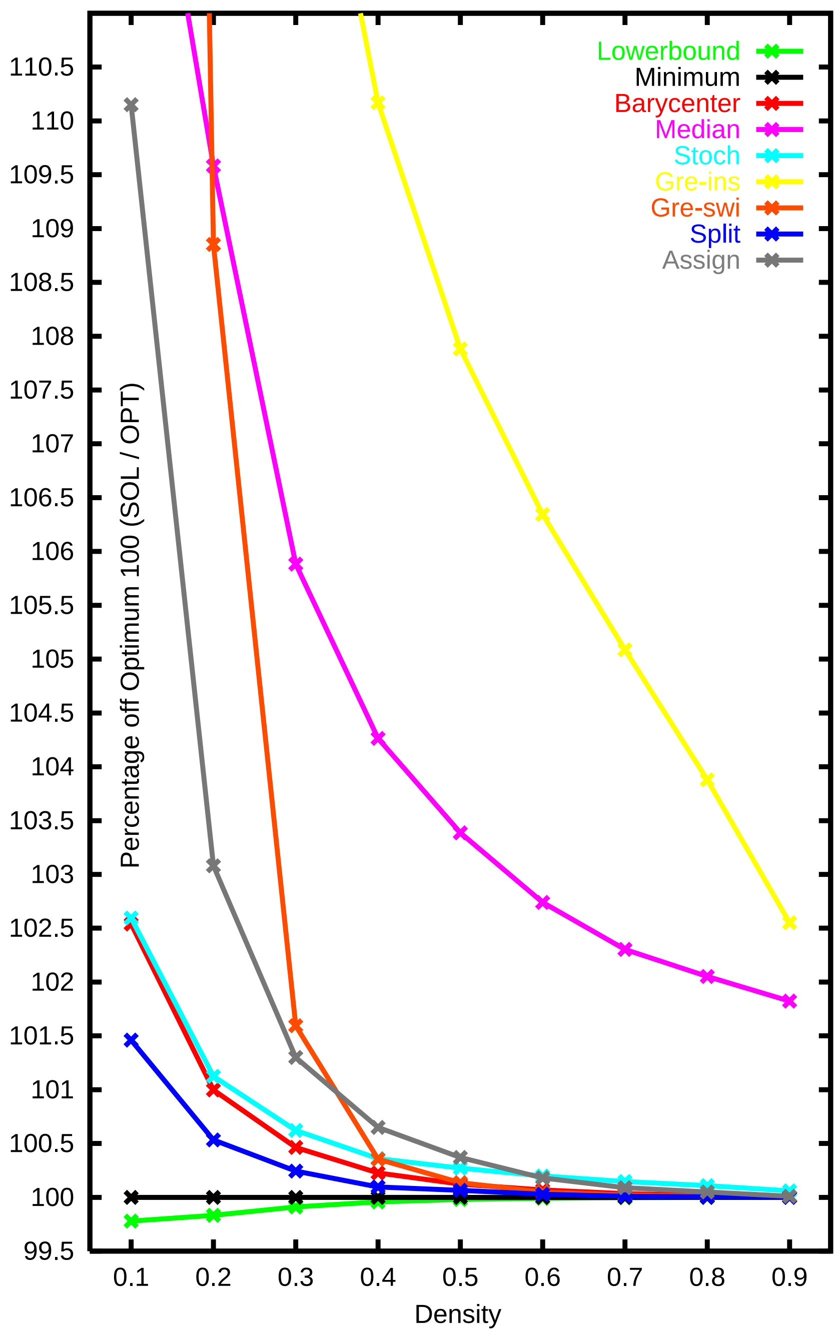

Figure 2: Results for 100 instances on $20 + 20$ nodes with increasing density

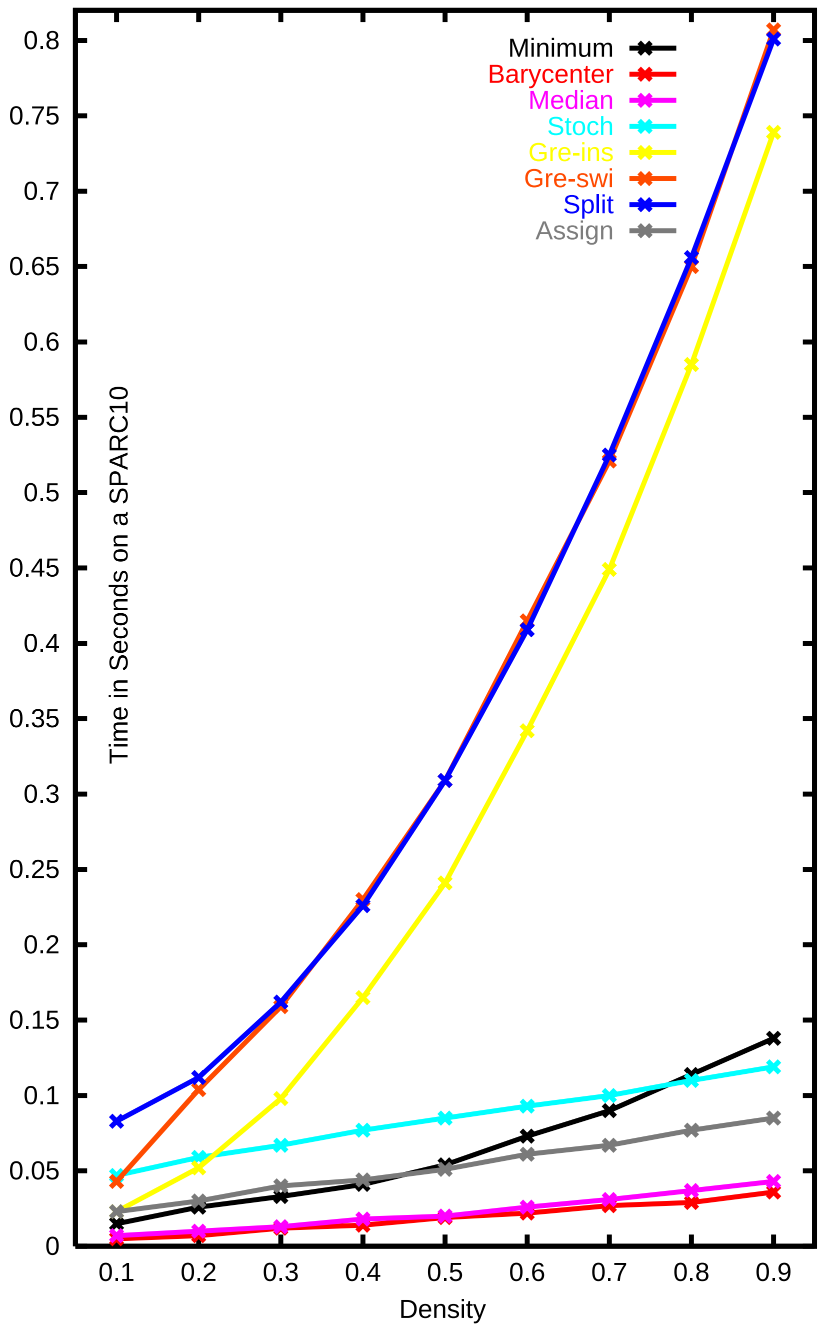

Figure 3: Time for 100 instances on 20 + 20 nodes with increasing density

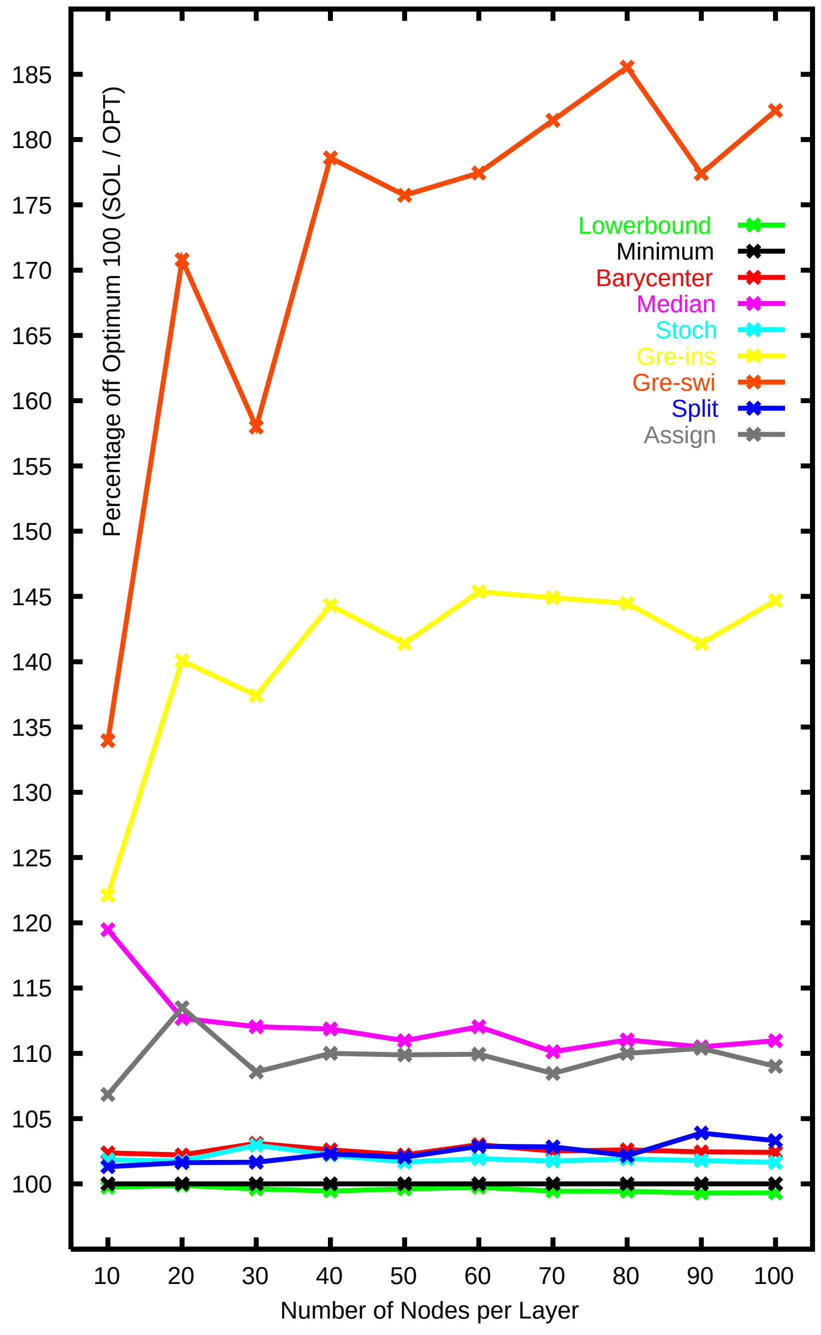

Figure 4: Results for 10 instances of sparse graphs with increasing size

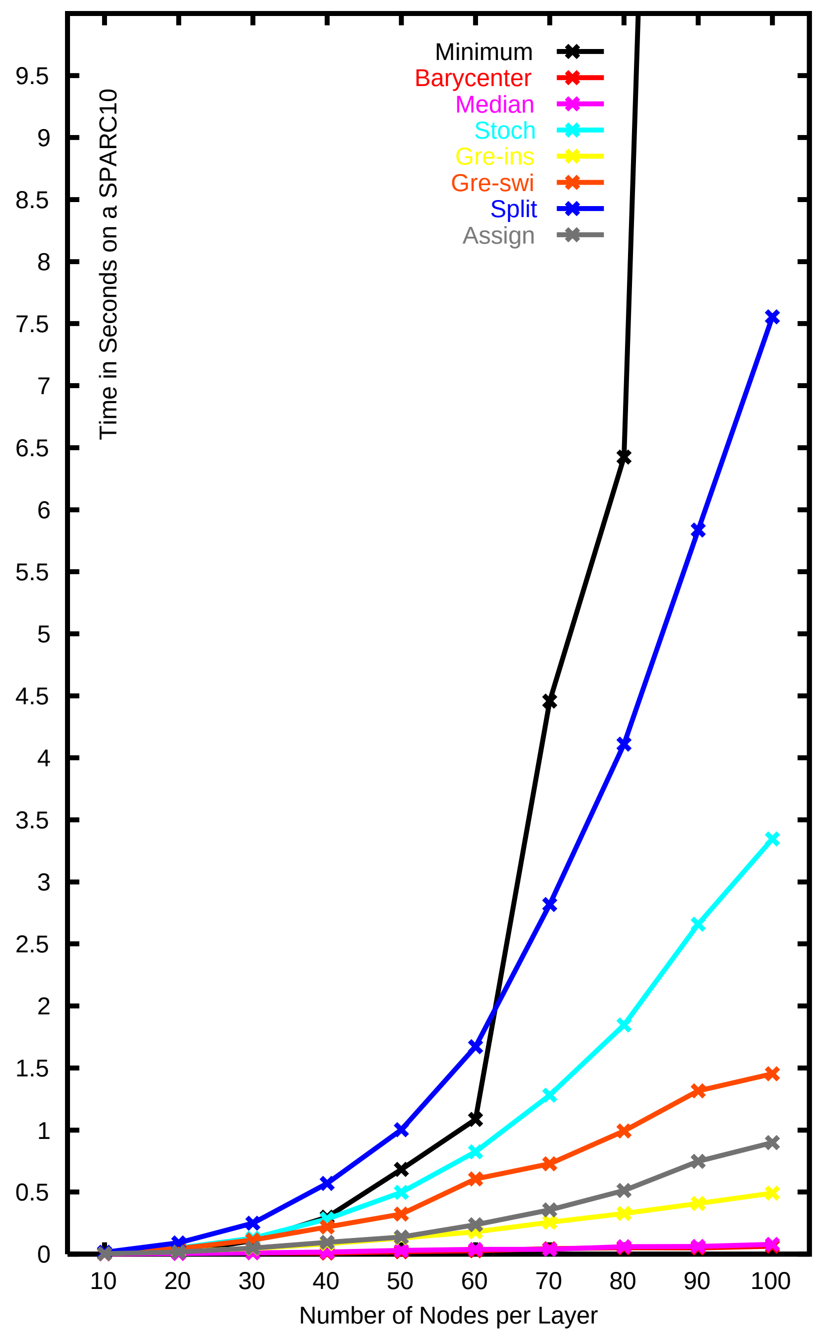

Figure 5: Time for 10 instances of sparse graphs with increasing size

In Figures 2 and 3, we give the results for “20+20-graphs”, i.e., bipartite graphs with 20 nodes on each layer and various fixed numbers of edges chosen uniformly and independently from the set of all possible edges. Each average is taken over 100 samples. The most surprising fact is perhaps that the exact computation by the branch and cut algorithm is faster than many of the heuristics. Only the barycenter and the median heuristic are between two to four times faster than the exact algorithm. The stochastic and assign heuristic take about the same time as the exact algorithm, whereas the split and the two greedy heuristics take much longer (see Fig. 3). The best results are obtained by the split heuristic. But also the results of the barycenter and the stochastic heuristic are quite good. For sparse graphs, the assign and the greedy-switch heuristic are quite far away from the optimum solution (10%, resp., 50%), whereas they achieve good solutions for dense graphs. However, in automatic graph drawing the graphs are usually sparse. The median heuristic is between 1% and 14 % away from the optimum solution. Greedy-insert shows the worst behaviour. Surprisingly, the lower bound is very close to the optimum solution, even in the sparse case (see Fig. 2).

In Figures 4 and 5, we concentrate on sparse instances in which, on the average, every node has two adjacent edges. We believe that such instances are among the most interesting in practical applications. It turns out that the stochastic, the split, and the barycenter heuristic perform very well in terms of quality (1%-4% off the optimum solution), however, the split heuristic takes roughly the same time as the branch and cut computation up to size 80+80, whereas the barycenter heuristic obtains results of similar quality as split, but much faster. The assign and the median heuristic are about 10% away from the optimum solution. Greedy-insert and greedy-switch behave worst for sparse graphs (see Fig. 4). For $n = 60$, the ranking of the heuristics with respect to increasing time is barycenter, median, greedy-insert, assign, greedy-switch, stochastic, exact, and split (see Fig. 5).

In Table 1, we repeat an experiment by Dresbach [4] for instances defined by Warfield [16] as follows: For $k = 3, 4, 5, 6, 7, 8$ we let $n_1 = k$, $n_2 = 2^k - 1$, and the adjacency matrix of the bipartite graph is a $n_1 \times n_2$ matrix whose rows are labelled $1, 2, \dots, k$, whose columns are labelled $1, 2, \dots, 2^k - 1$, and column j contains j in k-digit binary notation. Layer 1 is fixed and layer 2 is “optimized”. Again, it turns out that barycenter is the fastest method with excellent quality solutions. The results of the stochastic heuristic, the barycenter and the split heuristic are very close to the optimum solution. Up to size 7+127, the branch and cut algorithm needs only moderate computation time, for the instance 8+255 it is not competitive in terms of time, but we found it surprising that such a big linear ordering instance with $n = 255$ could be solved at all. The branch and cut algorithm was the only method that found the true optima for $k \geq 6$, whereas for $3 \leq k \leq 5$, the fact that the optimum value equals the value of the trivial lower bound seems to indicate that these instances are not hard.

n_1	n_2	m	Low	Min	Bary	Median	Stoch	Gre-ins	Gre-swi	Split	Assign
3	7	12	8	8	8	13	8	11	8	8	8
				100.00	100.00	162.50	100.00	137.50	100.00	100.00	100.00
				0.00	0.00	0.00	0.02	0.00	0.00	0.02	0.00
4	15	32	95	95	95	127	95	122	98	95	101
				100.00	100.00	133.68	100.00	128.42	103.16	100.00	106.32
				0.00	0.00	0.00	0.03	0.02	0.05	0.07	0.02
5	31	80	756	756	758	922	756	934	804	760	780
				100.00	100.27	121.96	100.00	123.55	106.35	100.53	103.17
				0.03	0.00	0.03	0.18	0.08	0.40	0.43	0.08
6	63	192	4998	5002	5015	5818	5004	6023	5523	5043	5120
				100.00	100.26	116.31	100.04	120.41	110.42	100.90	102.36
				0.73	0.05	0.07	1.38	0.38	2.87	2.65	0.38
7	127	448	29745	29778	29883	33641	29841	35152	34366	30086	30386
				100.00	100.35	112.97	100.21	118.05	115.41	101.03	102.04
				20.50	0.17	0.20	9.02	1.98	20.20	24.30	2.18
8	255	1024	165375	165602	166098	183342	165824	192633	202957	167546	168056
				100.00	100.30	110.71	100.13	116.32	122.56	101.17	101.48
				7200.00	0.95	1.08	67.90	7.33	147.00	189.00	21.50

Table 1: Results for Dresbach instances

4 Two Sided Crossing Minimization

The trivial lower bound on the number of crossings that turned out to be excellent in our previous experiments, can obviously be adapted to partial orderings rather than complete orderings (permutations) on one of the layers. This encouraged us to devise a simple branch and bound algorithm for the general two layer straightline crossing minimization problem in which both π_1 and π_2 must be determined. Namely, we enumerate all permutations π_1 (without loss of generality we can assume $|V_1| \leq |V_2|$, $V_1 = \{1, 2, \ldots, n\}$) as follows: Initially all $v \in V_1$ are unfixed. At depth l in a depth-first-search, $l-1$ nodes of V_1 are fixed in positions $1, 2, \ldots, l-1$. Then the first unfixed node in the canonical ordering of V_1 is fixed at position l, and the trivial lower bound L is computed for the resulting partial ordering. If L is greater than the value of the best known solution, the next unfixed node in the canonical ordering of V_1 is fixed at position l, else we move to position $l+1$, if $l < n$, and otherwise ($l = n$) we call the branch and cut algorithm to determine an optimum ordering of V_2 and update the best known solution, if necessary. Backtracking, i.e., moving from position l to position $l-1$ occurs whenever the list of unfixed nodes at depth l in the enumeration tree is exhausted. Before the enumeration is entered, a heuristic solution is determined in order to initialize the best known solution. A good initial solution makes the enumeration tree smaller.

We use this algorithm to determine optimum solutions for 10+10 graphs with increasing edge densities, 100 samples for each type of graph. All heuristics are iterated between the two layers until a local optimum is obtained, as outlined in the introduction, starting from the canonical ordering on V_1. An additional column labelled "LR-Opt" gives according results for the iterated minimum crossing computation by branch and cut, which is, remarkably, sometimes outperformed by the best iterated heuristics. For sparse instances, the minimum is much better than any of the heuristically found solutions (see Figure 7). The best heuristics barycenter, median, and LR-opt are very far away from the optimum solution. For density 0.1 the number of crossings is between 5 times and 33 times higher than the minimum straightline crossing number. For density 0.2 the best solution is still 60% away from the optimum. The ranking of the heuristics here is barycenter, LR-opt, split, median, stochastic, greedy-switch, assign, greedy-insert. The rank of greedy-switch improves for dense graphs. It turns out that with increasing density, the computation times increase rapidly for the minimum computation, whereas the heuristics are not very sensitive to density. The running times for the heuristics stay under 0.4 seconds, whereas the computation by the exact algorithm increased from 1.1 seconds for density 0.1 to 1550 seconds for density 0.9.

In Figure 6, we show an example of a 10+10 graph with 20 edges. The first drawing was found by the LR-opt heuristic and has 30 crossings, the second by the barycenter heuristic and contains 10 crossings and the third one is the optimum solution with only 4 crossings.

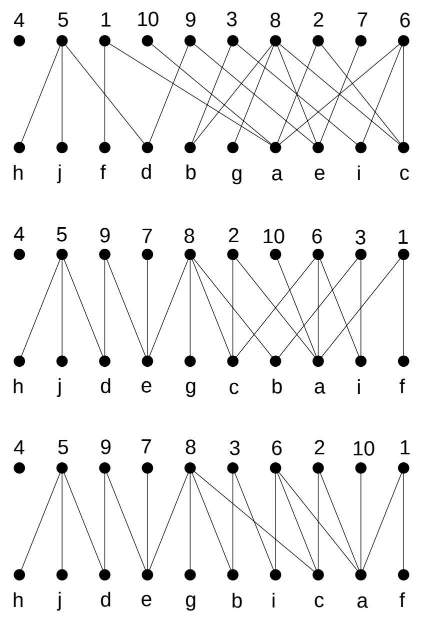

Figure 6: Time for 10 instances of sparse graphs with increasing size

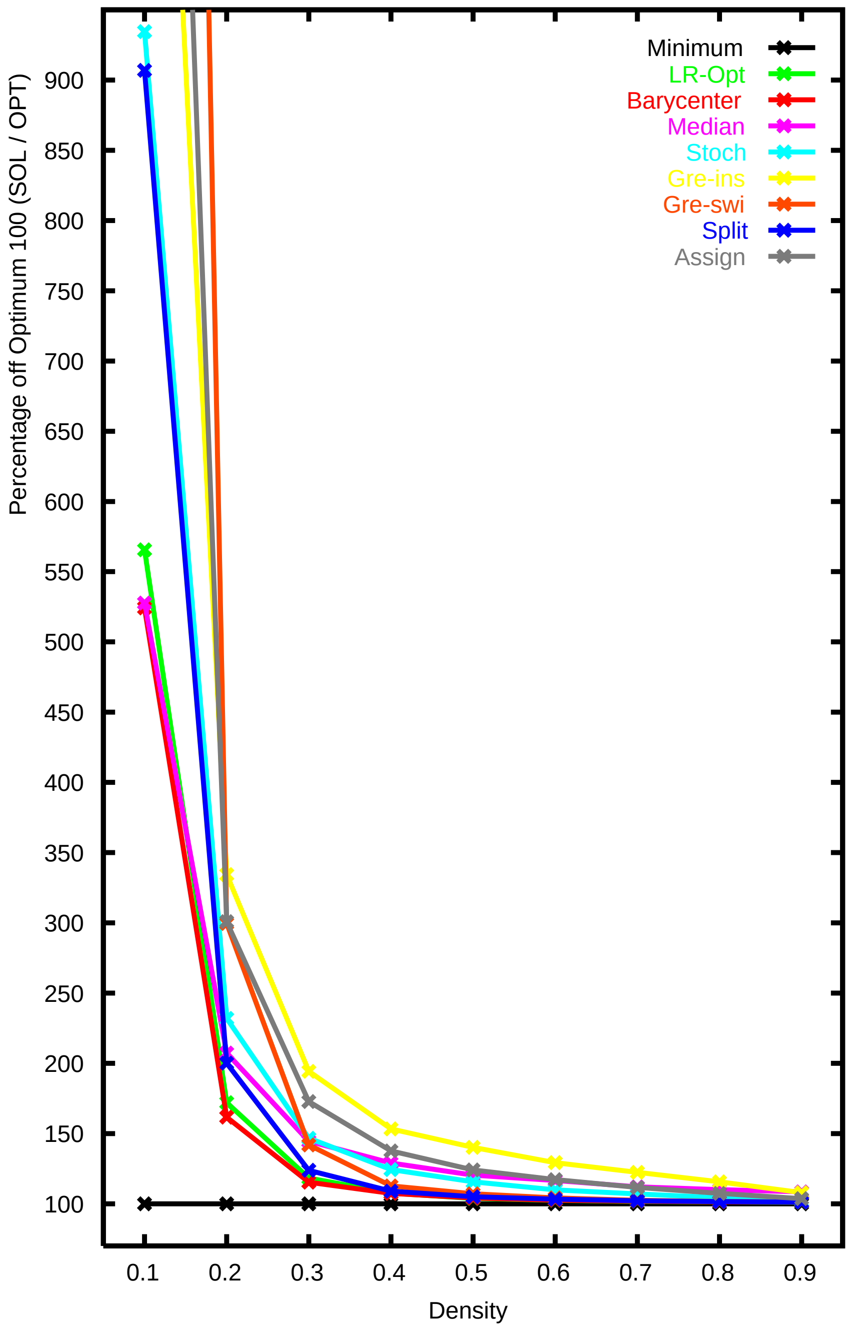

Figure 7: Results for 100 instances on $10 + 10$ nodes with one trial

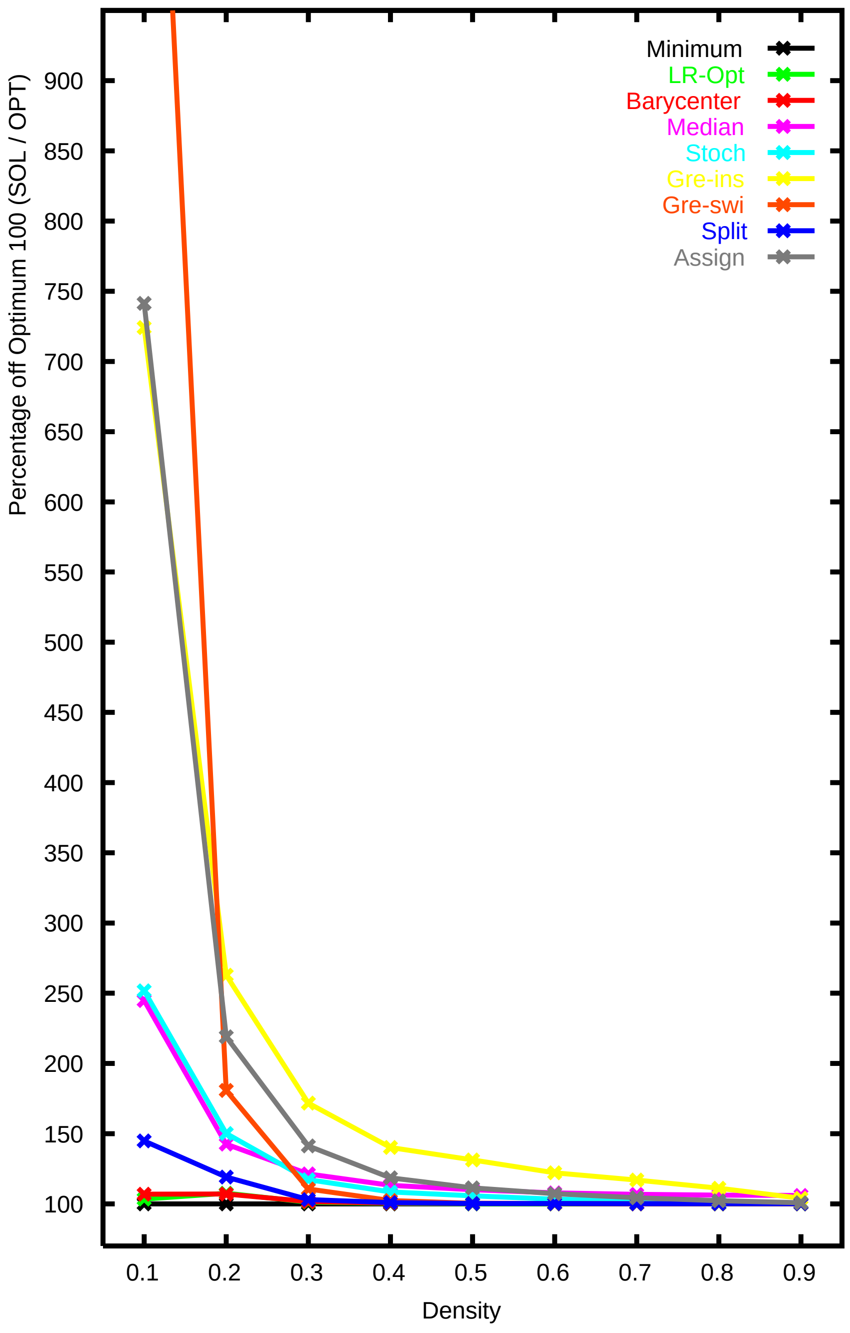

Figure 8: Results for 100 instances on $10 + 10$ nodes with 10 trials

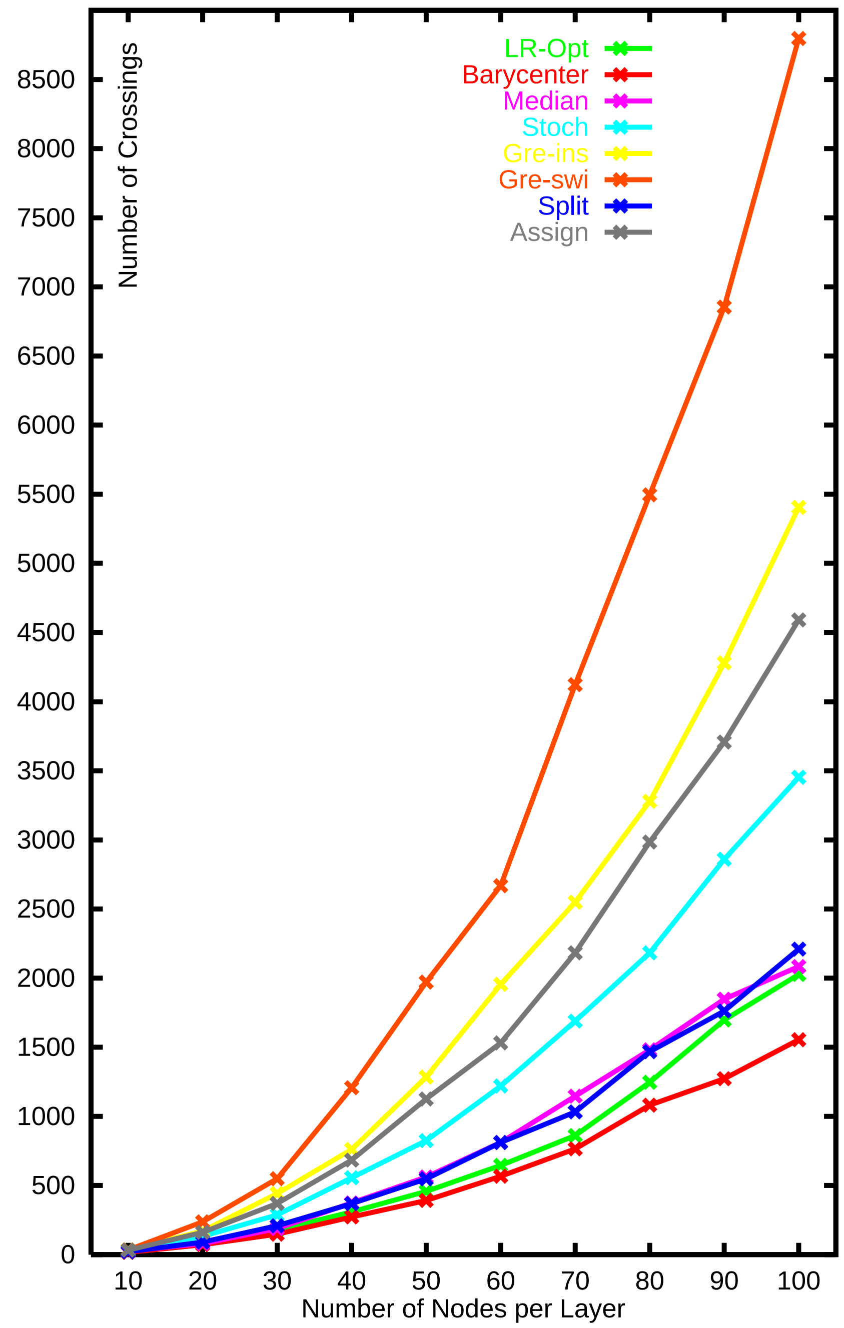

Figure 9: Results for 10 instances on sparse graphs (absolute)

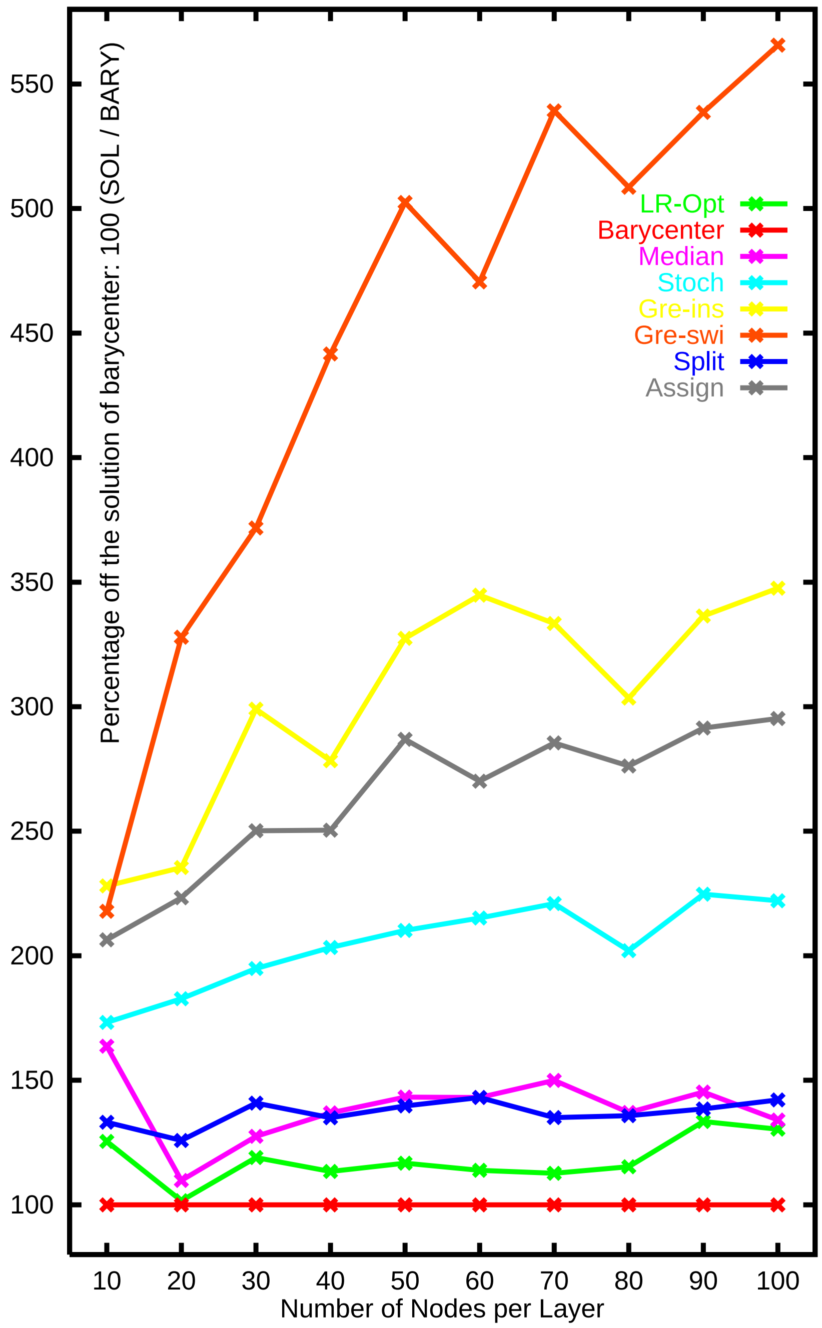

Figure 10: Results for 10 instances on sparse graphs (relative to barycenter)

Within one hour of computation time, we can find optimum solutions for 11+11 instances with up to 80% density, 12+12 with up to 50% density, 13+13 with up to 30% density, 14+14, 15+15, 16+16 with up to 10% density.

In Figure 8, we repeat the same experiment with 10 starts from random orderings of the nodes in V_1 and take the best solutions found. The results show that a considerable performance gain for all heuristics can be achieved. LR-Opt, barycenter and split obtain results of similar good quality. But still, for sparse graphs, they are at least 7% away from the optimum, split even 44%.

Figures 9 and 10 deal with the more interesting sparse instances of bigger size for which we can not compute the optimum anymore. Thus, we divided the number of crossings found by the heuristics (shown in Figure 9) by that computed by barycenter (see Figure 10). This allows us to compare the behaviour of the heuristics for one layer fixed against the free case. Barycenter constantly gives the best solutions for sparse graphs. Also, median is, contrary to the one-layer fixed case, among the best heuristics. Barycenter, LR-opt, median and split give the best solutions. However, we do not know how far their solutions are away from the optimum. Stochastic behaves worse for two free layers, although it was originally designed for this problem. Assign and greedy-insert are among the worst heuristics, but their quality seems to stay constant with increasing number of nodes, in contrary to greedy-switch. With 10 different starts from random orderings of the nodes in V_1, the quality of the results improves only slightly. The data of all of our experiments is given in the Appendix. Summarizing, the barycenter method turns out to be the clear winner, both in terms of quality as well as in terms of computation time.

5 Conclusions

The outcome of our computational experiments lead to the following conclusions.

(1) When one layer is fixed, and the free layer does not contain more than 60 vertices, which is well beyond typical practical instance sizes, the exact minimum crossing number can be efficiently computed in practice, so there is no real need for heuristics.

(2) In the general case, small sparse instances, which often occur in applications, can be solved to optimality if the smaller sized shore has up to about 15 vertices. For larger instances, the iterated barycenter method, started with a few random orderings of one layer, is clearly the method of choice among all tested methods.

Acknowledgements

We would like to thank Thomas Ziegler for implementing and running all heuristics in this paper except LR-opt and Stefan Dresbach for helpful discussions concerning the stochastic heuristic.

References

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

[2] C. Catarci. The assignment heuristic for crossing reduction. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(3), 1995.

[3] CPLEX Optimization Inc. *Using the CPLEX callable library and the CPLEX mixed integer library*, 1993.

[4] S. Dresbach. A new heuristic layout algorithm for DAGs. In U. Derigs and A. B. . A. Drexl, editors, *Operations Research Proceedings 1994*, pages 121–126. Springer Verlag, Berlin, 1994.

[5] S. Dresbach, 1995. Personal communication.

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

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

[8] P. Eades and N. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 10:379–403, 1994.

[9] M. Garey and D. Johnson. Crossing number is NP-complete. *SIAM J. Algebraic Discrete Methods*, 4:312–316, 1983.

[10] M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32:1195–1220, 1984.

[11] M. Grötschel, M. Jünger, and G. Reinelt. Optimal triangulation of large real world input-output matrices. *Statistische Hefte*, 25:261–295, 1984.

[12] M. Grötschel, M. Jünger, and G. Reinelt. Facets of the linear ordering polytope. *Mathematical Programming*, 33:43–60, 1985.

[13] D. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing.* ACM Press, Addison-Wesley Publishing Company, New York, 1993.

[14] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical systems. *IEEE Trans. Syst. Man Cybern.*, SMC-11(2):109–125, 1981.

[15] V. Valls, R. Marti, and P. Lino. A branch and bound algorithm for minimizing the number of crossing arcs in bipartite graphs. *Journal of Operational Research*, 90:303–319, 1996.

[16] J. Warfield. Crossing theory and hierarchy mapping. *IEEE Trans. Syst. Man Cybern.*, SMC-7(7):502–523, 1977.

6 Appendix: Tables

The tables show the average number of crossings taken over all sampled instances of this type, the relative size of this number in percentage of the minimum number of crossings, and the average running time in seconds on a SUN Sparcstation 10 for the investigated heuristics and exact algorithms.

n_i	m	Low	Min	Bary	Median	Stoch	Gre-ins	Gre-swi	Split	Assign
20	40	180.35	180.75	185.34	206.27	185.44	248.37	275.99	183.39	199.09
		99.78	100.00	102.54	114.12	102.60	137.41	152.69	101.46	110.15
			0.02	0.01	0.01	0.05	0.02	0.04	0.08	0.02
20	80	957.62	959.23	968.80	1051.14	970.01	1175.11	1044.14	964.35	988.76
		99.83	100.00	101.00	109.58	101.12	122.51	108.85	100.53	103.08
			0.03	0.01	0.01	0.06	0.05	0.10	0.11	0.03
20	120	2420.14	2422.32	2433.53	2564.82	2437.39	2763.72	2460.94	2428.23	2453.89
		99.91	100.00	100.46	105.88	100.62	114.09	101.59	100.24	101.30
			0.03	0.01	0.01	0.07	0.10	0.16	0.16	0.04
20	160	4625.79	4627.72	4638.24	4825.06	4644.35	5098.27	4644.10	4632.17	4657.85
		99.96	100.00	100.23	104.26	100.36	110.17	100.35	100.10	100.65
			0.04	0.01	0.02	0.08	0.17	0.23	0.23	0.04
20	200	7560.42	7561.88	7571.08	7817.99	7582.47	8157.86	7572.24	7566.79	7589.64
		99.98	100.00	100.12	103.39	100.27	107.88	100.14	100.07	100.37
			0.05	0.02	0.02	0.09	0.24	0.31	0.31	0.05
20	240	11314.37	11315.55	11323.26	11625.54	11338.06	12033.34	11321.10	11318.68	11336.09
		99.99	100.00	100.07	102.74	100.20	106.34	100.05	100.03	100.18
			0.07	0.02	0.03	0.09	0.34	0.42	0.41	0.06
20	280	15859.70	15860.35	15865.69	16225.57	15883.69	16667.12	15863.66	15861.76	15874.86
		99.99	100.00	100.03	102.30	100.15	105.09	100.02	100.01	100.09
			0.09	0.03	0.03	0.10	0.45	0.52	0.53	0.07
20	320	21290.56	21290.76	21294.12	21727.43	21313.78	22116.56	21292.93	21291.56	21300.43
		99.99	100.00	100.02	102.05	100.12	103.88	100.01	100.00	100.05
			0.11	0.03	0.04	0.11	0.59	0.65	0.66	0.08
20	360	27751.63	27751.69	27752.99	28257.47	27768.41	28459.57	27752.01	27751.84	27754.31
		100.00	100.00	100.01	101.82	100.06	102.55	100.00	100.00	100.01
			0.14	0.04	0.04	0.12	0.74	0.81	0.80	0.09

Table 2: Results for 100 instances of the one sided crossing minimization problem on $20 + 20$ nodes with increasing density (see Figs. 2 and 3)

n_i	m	Low	Min	Bary	Median	Stoch	Gre-ins	Gre-swi	Split	Assign
10	20	37.90	38.00	38.90	45.40	38.70	46.40	50.90	38.50	40.60
		99.74	100.00	102.37	119.47	101.84	122.11	133.94	101.32	106.84
			0.00	0.00	0.00	0.01	0.00	0.01	0.02	0.00
20	40	171.70	171.90	175.70	193.70	174.90	240.80	293.60	174.70	195.10
		99.88	100.00	102.21	112.68	101.74	140.08	170.80	101.63	113.50
			0.01	0.01	0.01	0.05	0.02	0.05	0.09	0.02
30	60	436.60	438.30	451.90	491.10	451.30	602.30	692.40	445.60	475.90
		99.61	100.00	103.10	112.05	102.97	137.42	157.97	101.67	108.58
			0.11	0.01	0.01	0.13	0.05	0.11	0.25	0.05
40	80	761.50	765.70	785.60	856.60	782.70	1105.00	1367.50	783.20	842.30
		99.45	100.00	102.60	111.87	102.22	144.31	178.60	102.29	110.00
			0.30	0.01	0.02	0.28	0.08	0.22	0.57	0.09
50	100	1247.30	1252.20	1279.90	1389.50	1273.20	1770.60	2200.50	1277.80	1375.90
		99.61	100.00	102.21	110.97	101.68	141.40	175.73	102.04	109.88
			0.68	0.02	0.03	0.50	0.13	0.32	1.00	0.14
60	120	1683.10	1687.60	1738.30	1890.90	1720.20	2453.10	2994.50	1736.10	1855.30
		99.73	100.00	103.00	112.05	101.93	145.36	177.44	102.87	109.94
			1.09	0.03	0.04	0.83	0.18	0.61	1.67	0.24
70	140	2465.00	2479.00	2541.30	2730.00	2522.50	3592.20	4498.80	2549.20	2688.60
		99.44	100.00	102.51	110.13	101.76	144.91	181.48	102.83	108.46
			4.46	0.04	0.04	1.28	0.26	0.73	2.82	0.36
80	160	3153.90	3172.10	3254.60	3521.60	3232.90	4583.10	5885.70	3240.60	3488.90
		99.43	100.00	102.60	111.02	101.92	144.48	185.55	102.16	109.99
			6.42	0.05	0.06	1.85	0.33	0.99	4.11	0.51
90	180	4104.00	4132.80	4233.70	4566.80	4206.80	5843.70	7331.30	4293.90	4561.60
		99.30	100.00	102.44	110.50	101.79	141.40	177.39	103.90	110.38
			25.13	0.05	0.06	2.66	0.41	1.32	5.84	0.75
100	200	5127.40	5162.70	5287.50	5728.80	5247.60	7469.90	9407.50	5333.50	5627.50
		99.32	100.00	102.42	110.97	101.64	144.69	182.22	103.31	109.00
			435.51	0.06	0.08	3.35	0.49	1.45	7.56	0.90

Table 3: Results for 10 instances of the one sided crossing minimization problem of sparse graphs with increasing size (see Figs. 4 and 5)

n_i	m	Min	LR-Opt	Bary	Median	Stoch	Gre-ins	Gre-swi	Split	Assign
10	10	0.29	1.64	1.52	1.53	2.71	4.32	9.61	2.63	5.42
		100.00	565.52	524.14	527.59	934.48	1489.66	3313.79	906.90	1868.97
		1.10	0.01	0.01	0.01	0.03	0.02	0.02	0.04	0.01
10	20	11.62	19.99	18.78	24.08	26.96	38.85	34.81	23.25	34.96
		100.00	172.03	161.62	207.23	232.01	334.34	299.57	200.09	300.86
		3.89	0.02	0.01	0.01	0.06	0.04	0.03	0.07	0.02
10	30	56.60	66.98	65.30	81.78	82.98	109.96	80.29	70.11	97.80
		100.00	118.34	115.37	144.49	146.61	194.28	141.86	123.87	172.79
		14.06	0.02	0.02	0.02	0.07	0.06	0.07	0.11	0.02
10	40	146.89	157.91	157.70	189.55	182.77	225.26	165.65	160.20	202.10
		100.00	107.50	107.36	129.04	124.43	153.35	112.77	109.06	137.59
		43.02	0.03	0.02	0.02	0.08	0.10	0.11	0.15	0.03
10	50	276.78	287.32	288.15	333.25	320.21	387.87	296.38	290.79	343.65
		100.00	103.81	104.11	120.40	115.69	140.14	107.08	105.06	124.16
		91.58	0.04	0.03	0.02	0.09	0.13	0.15	0.21	0.03
10	60	463.17	475.04	475.52	539.59	509.38	598.98	482.76	478.46	542.88
		100.00	102.56	102.67	116.50	109.98	129.32	104.23	103.30	117.21
		206.61	0.06	0.03	0.03	0.10	0.17	0.22	0.28	0.03
10	70	698.35	709.91	710.88	782.33	747.20	854.61	715.73	712.73	779.79
		100.00	101.66	101.79	112.03	107.00	122.38	102.49	102.06	111.67
		379.12	0.07	0.04	0.03	0.11	0.22	0.29	0.35	0.04
10	80	1008.38	1021.46	1021.44	1110.39	1051.66	1165.97	1025.84	1024.78	1083.39
		100.00	101.30	101.30	110.12	104.29	115.63	101.73	101.63	107.44
		763.53	0.08	0.04	0.03	0.12	0.27	0.34	0.40	0.04
10	90	1405.57	1420.68	1421.86	1524.18	1430.86	1516.62	1423.90	1421.72	1456.97
		100.00	101.08	101.16	108.44	101.80	107.90	101.30	101.15	103.66
		1549.12	0.07	0.03	0.03	0.12	0.29	0.32	0.37	0.04

Table 4: Results for 100 instances of the two sided crossing minimization problem on $10 + 10$ nodes with increasing density (see Figs. 7 and 8)

n_i	m	Min	LR-Opt	Bary	Median	Stoch	Gre-ins	Gre-swi	Split	Assign
10	10	0.29	0.30	0.31	0.71	0.73	2.10	3.95	0.42	2.15
		100.00	103.45	106.90	244.83	251.72	724.14	1362.07	144.83	741.38
		1.10	0.11	0.08	0.08	0.27	0.21	0.16	0.38	0.10
10	20	11.62	12.50	12.44	16.57	17.44	30.55	21.00	13.83	25.44
		100.00	107.57	107.06	142.60	150.09	262.91	180.72	119.02	218.93
		3.89	0.18	0.12	0.13	0.52	0.38	0.34	0.64	0.17
10	30	56.60	57.27	57.46	68.66	66.33	97.22	62.59	58.30	79.97
		100.00	101.18	101.52	121.31	117.19	171.77	110.58	103.00	141.29
		14.06	0.26	0.17	0.15	0.68	0.60	0.62	1.01	0.23
10	40	146.89	147.35	147.73	166.41	159.31	205.97	150.34	148.24	174.12
		100.00	100.31	100.57	113.29	108.46	140.22	102.35	100.92	118.54
		43.02	0.36	0.21	0.18	0.79	0.90	1.02	1.45	0.26
10	50	276.78	277.11	277.78	304.62	292.34	363.43	277.85	277.61	308.26
		100.00	100.12	100.36	110.06	105.62	131.31	100.39	100.30	111.37
		91.58	0.47	0.26	0.22	0.87	1.23	1.50	2.03	0.30
10	60	463.17	463.76	464.07	499.41	478.48	565.63	464.54	464.17	497.17
		100.00	100.13	100.19	107.82	103.31	122.12	100.30	100.22	107.34
		206.61	0.59	0.32	0.25	0.96	1.65	2.15	2.67	0.34
10	70	698.35	698.75	699.23	745.00	712.78	816.80	699.37	699.04	728.95
		100.00	100.06	100.13	106.68	102.07	116.96	100.15	100.10	104.38
		379.12	0.68	0.34	0.29	1.03	2.23	2.78	3.30	0.37
10	80	1008.38	1008.62	1008.88	1070.82	1018.66	1120.31	1008.96	1008.94	1031.45
		100.00	100.02	100.05	106.19	101.02	111.10	100.06	100.06	102.29
		763.53	0.81	0.37	0.31	1.11	2.70	3.39	3.89	0.41
10	90	1405.57	1406.14	1406.22	1490.03	1410.31	1461.52	1406.43	1406.44	1416.64
		100.00	100.04	100.05	106.01	100.34	103.98	100.06	100.06	100.79
		1549.12	0.70	0.33	0.34	1.17	2.86	3.13	3.53	0.44

Table 5: Results for 100 instances of the two sided crossing minimization problem on 10 + 10 nodes with increasing density, 10 trials each (see Figs. 7 and 8)

n_i	m	LR-Opt	Bary	Median	Stoch	Gre-ins	Gre-swi	Split	Assign
10	20	19.70	15.70	25.70	27.20	35.80	34.20	20.90	32.40
		0.02	0.02	0.01	0.05	0.04	0.04	0.06	0.02
20	40	73.70	72.50	79.60	132.50	170.70	237.70	91.20	161.90
		0.10	0.03	0.04	0.36	0.17	0.17	0.41	0.06
30	60	176.00	147.90	188.50	288.20	442.30	549.80	208.30	370.00
		0.48	0.10	0.09	1.18	0.49	0.48	1.33	0.15
40	80	309.80	273.30	374.20	555.70	760.60	1207.00	368.80	684.40
		1.81	0.17	0.14	2.72	0.93	0.67	3.45	0.28
50	100	457.70	392.30	561.90	824.40	1284.40	1971.20	548.10	1125.60
		5.87	0.25	0.17	5.92	1.37	1.10	7.14	0.47
60	120	645.60	567.00	811.20	1219.90	1954.80	2667.90	811.10	1531.20
		13.34	0.38	0.24	8.58	2.24	1.87	10.52	0.73
70	140	861.30	764.60	1146.20	1689.30	2549.30	4122.80	1032.40	2182.40
		24.95	0.55	0.34	14.09	2.89	2.19	19.48	1.02
80	160	1246.10	1080.70	1481.30	2183.30	3279.40	5495.90	1467.70	2984.90
		62.65	0.68	0.52	21.09	4.58	3.22	25.01	1.46
90	180	1697.70	1272.40	1848.00	2859.50	4280.00	6853.70	1762.40	3708.20
		86.37	1.10	0.57	31.84	6.41	4.30	38.36	2.28
100	200	2027.30	1555.10	2084.10	3453.10	5405.00	8796.30	2209.40	4591.00
		178.93	1.46	0.82	40.23	7.41	5.25	47.78	2.67

Table 6: Results for 10 instances of the two sided crossing minimization problem of sparse graphs (see Figs. 9 and 10)

n_i	m	LR-Opt	Bary	Median	Stoch	Gre-ins	Gre-swi	Split	Assign
10	20	13.60	12.70	18.70	17.50	30.00	22.30	14.70	25.70
		0.12	0.15	0.12	0.55	0.40	0.34	0.68	0.16
20	40	51.00	48.30	59.10	89.00	150.80	163.40	63.70	128.60
		0.98	0.42	0.39	3.61	1.82	1.58	3.93	0.68
30	60	133.40	117.00	145.80	228.60	421.30	422.10	160.10	321.80
		5.55	0.96	0.76	11.48	4.59	4.18	13.13	1.42
40	80	234.10	212.40	271.40	432.80	724.50	949.80	279.90	589.70
		18.45	1.75	1.29	26.57	8.25	7.42	31.26	6.91
50	100	384.20	325.60	407.30	715.60	1245.60	1715.90	462.70	966.20
		52.01	2.79	2.06	51.33	13.59	11.60	60.80	4.83
60	120	541.10	479.90	599.90	1106.80	1909.70	2472.10	654.00	1425.80
		128.12	4.38	2.93	92.08	21.97	18.27	114.07	7.14
70	140	733.20	641.30	858.00	1489.30	2514.30	3640.00	896.90	1973.20
		304.08	5.79	3.82	139.95	30.18	23.22	175.83	11.04
80	160	1022.90	903.70	1145.10	1993.30	3248.70	4843.50	1169.60	2634.00
		619.36	7.57	5.28	204.64	38.96	31.18	264.82	14.59
90	180	1282.50	1044.70	1323.70	2516.50	4209.10	6228.20	1466.40	3289.20
		1134.67	10.81	6.55	307.44	57.13	43.19	377.72	20.20
100	200	1599.20	1313.20	1793.20	3119.40	5323.90	8145.30	1807.60	4165.10
		2313.48	13.76	8.02	402.74	67.13	50.24	504.25	27.12

Table 7: Results for 10 instances of the two sided crossing minimization problem of sparse graphs, 10 trials each