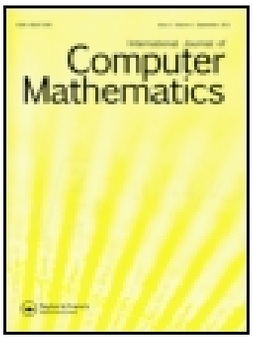

International Journal of Computer Mathematics

ISSN: 0020-7160 (Print) 1029-0265 (Online) Journal homepage: https://www.tandfonline.com/loi/gcom20

Experiments on drawing 2-level hierarchical graphs

Erkki Mäkinen

To cite this article: Erkki Mäkinen (1990) Experiments on drawing 2-level hierarchical graphs, International Journal of Computer Mathematics, 36:3-4, 175-181, DOI: 10.1080/00207169008803921

To link to this article: https://doi.org/10.1080/00207169008803921

Intern. J. Computer Math., Vol. 36, pp. 175–181
Reprints available directly from the publisher
Photocopying permitted by license only

EXPERIMENTS ON DRAWING 2-LEVEL HIERARCHICAL GRAPHS

ERKKI MÄKINEN

University of Tampere, Department of Computer Science, P.O. Box 607, SF-33101 Tampere, Finland

(*Received 26 March 1990*)

This paper studies different heuristics for drawing 2-level hierarchical graphs. Especially, we compare the barycenter and the median heuristics. We show that the barycenter heuristic clearly outperforms the median heuristic, although only the latter has a proved bound for the maximum error done when two vertices are ordered. Moreover, we improve a known heuristic, called the greedy switching, by introducing the barycenter heuristic as a preprocessing phase for it.

KEY WORDS: Graph drawing, edge crossing, hierarchical graph, barycenter heuristic, median heuristic, greedy switching.

C.R. CATEGORIES: F.2.2, G.2.2.

1. INTRODUCTION

Several software and other applications use algorithms for drawing graphs. Such applications include graphical editors and browsers [10], algorithm animation systems [3, 9], automatic tools for designing different kind of diagrams [1, 2] and various applications outside the boundaries of computer science [12, 13].

None of the drawing methods so far presented in the literature produces pleasing drawings for all applications and for all input graphs. It is presumable that such an algorithm will never be presented. This motivates the study of graph drawing in different subcases. Drawing hierarchical graphs is one of the most extensively studied subcases. It is also the subject of the present paper.

A (*proper*) *k-level hierarchical graph* $G=(V_1 \cup \cdots \cup V_k, E)$, $V_i \cap V_j = \emptyset$, $i \neq j$, has the characteristic feature that each edge in E connects vertices from consecutive subsets V_i and V_{i+1}. A k-level hierarchical graph is drawn so that the vertices in each subset V_i are placed to a line, called a *level*. The levels are drawn into ascending order of the corresponding subset indices and the edges are drawn by using straight lines. The problem is to permute the vertices in each level so that the number of edge crossings is minimized.

A natural heuristic approach to the k-level drawing problem is to divide it to $k-1$ problems of drawing a 2-level hierarchical graph, i.e. a bipartite graph. Once we have drawn one of the bipartite graphs we have $k-2$ simpler problems left. In each of these simpler problems we have to permute the vertices in one of the levels of a bipartite graph while the other level is fixed. This problem is referred to as the

level permutation problem (LPP). The problem of permuting both levels of a bipartite graph is referred to as the *bipartite drawing problem* (BDP).

Hence, in BDP we are given a bipartite graph $G=(V_1 \cup V_2, E)$ and we must find linear orders $f_1: V_1 \to \{1,2,\ldots,|V_1|\}$ and $f_2: V_2 \to \{1,2,\ldots,|V_2|\}$ of the vertex subsets minimizing the number of edge crossings in the corresponding drawing. On the other hand, an instance of LPP consists of a bipartite graph $G=(V_1 \cup V_2, E)$ and a linear order f_1 for V_1, and the problem is to find a linear order f_2 for V_2 minimizing the number of edge crossings.

Garey and Johnson [8] have shown that BDP is NP-complete by reducing the optimal linear arrangement problem to it. The reduction uses multi edges in BDP but it can be modified to use single edges only. Eades *et al.* [5] have also shown that LPP is NP-complete by reducing the feedback arc set problem to it.

This paper is devoted to different heuristics for LPP. These heuristics can be used for BDP, too. Namely, we can repeatedly change the roles of the levels and expect that the repeated orderings converge to a solution for BDP.

There are two basic categories of LPP heuristics. In chapters 2 and 3 we shall study *context-free* heuristics. These heuristics assign the same approximate value to each vertex irrespective of the other vertices of the graph in question. The solution is then obtained by ordering the vertices according to these values. On the other hand, *context-sensitive* heuristics first determine the exact numbers of edge crossings connected to the relative orders in question. Based on these numbers they then try to find a linear order which minimizes the number of edge crossings. Notice that if there is no circuit in the digraph representing the pairwise orders, context-sensitive heuristics determine the exact solution.

2. THE BARYCENTER AND THE MEDIAN HEURISTICS

We start by giving some notations. Let u and v be vertices on the same level. The number of crossings caused by the edges incident with u with the edges incident with v depends only on the relative order of u and v. Let c_{uv} denote the number of these crossings if u is on the left of v.

If $G=(V_1 \cup V_2, E)$ is the graph related to an instance of LPP, we suppose that the vertices in V_1 are numbered according to its linear order f_1, and the vertices in V_2 are numbered arbitrarily from 1 to $|V_2|$. The *neighborhood matrix* T related to G is a binary $|V_2| \times |V_1|$ matrix whose element $T[i,j]$ is 1 if and only if u_i in V_2 is incident with u_j in V_1.

The *barycenter heuristic* (BC) [11] orders the vertices according to the average positions of the vertices incident with them. Hence, for each vertex u_i in V_2 we go through the ith row of T and calculate the arithmetic mean of the positions of 1's. More formally, we define

$$bc(u_i) = \sum_{j=1,\ldots,n} j \cdot T[i,j] \Big/ \sum_{j=1,\ldots,n} T[i,j], \text{ where } n = |V_1|.$$

The correctness of the BC heuristic is approximated in [11] by counting the

numbers of vertex pairs for which $bc(u) < bc(v)$ although $c_{uv} > c_{vu}$ or $c_{uv} = c_{vu}$. The percentage of these pairs is called the *inconsistence rate* of the BC heuristic [11]. This rate is 0 when $n \leqq 4$, and it is below 2.5 when $n \leqq 10$. However, the inconsistence rate as defined in [11] does not count all erroneous behavior of the BC heuristic. It is injurious for the ordering process if we have $bc(u) = bc(v)$ in a case where c_{uv} and c_{vu} actually are unequal. Namely, when $bc(u) = bc(v)$, we must either order u and v arbitrarily or use some additional heuristic for the ordering. Thus, when we in chapter 3 empirically study the correctness of the BC heuristic, we allow $bc(u) < bc(v)$ and $bc(u) > bc(v)$ if $c_{uv} = c_{vu}$, but we do not allow $bc(u) = bc(v)$ if $c_{uv} < c_{vu}$ or $c_{uv} > c_{vu}$. The BC heuristic is called *averaging* in [4].

If $y_1, y_2, \ldots, y_k$ are the positions of 1-bits in the ith row of the neighborhood matrix T, then the *median heuristic* (ME) [6] assigns $f_2(u)$ to be y_m, where $m = \lceil k/2 \rceil$ for each u in V_2. This is denoted by $me(u)$. No empirical evidence for the favour of the median heuristic is yet presented in the literature. However, Eades and Wormald [6] have shown that the number of edge crossings in the drawings produced by this heuristic is never more than three times larger than in the optimal drawing. This follows from their lemma which states that for all pairs of vertices u and v, $me(u) \leqq me(v)$ implies $c_{uv} \leqq 3c_{vu}$. Eades and Wormald [6] have also shown that the bound 3 is optimal for a large family of heuristics. The following theorem indicates that the situation is completely different when the BC heuristic is concerned.

THEOREM 1 *For each integer $d, d > 0$, there are bipartite graphs having vertices u and v such that $bc(u) < bc(v)$ and $d \cdot c_{vu} < c_{uv}$.*

Proof Let d be any positive integer. Define now a bipartite graph $G = (V_1 \cup V_2, E)$ such that V_2 contains vertices u and v having the following properties. Vertex u is incident with the $(d+2)$th vertex in the linear order of V_1. Vertex v is incident with the vertices $1, 2, \ldots, d+1$ and $(d+2)^2$ in the linear order of V_1. Now $bc(u) < bc(v)$ although $c_{uv} = d+2$ and $c_{vu} = 1$. ∎

Because of the large number of vertices needed in V_1 to make the construction of the proof above possible, Theorem 1 is by no means fatal for the use of the BC heuristic in practice. In most applications the number of vertices in a level of a bipartite graph can well be supposed to be no more than 100.

3. BR vs. ME

In this chapter we empirically compare the BR and ME heuristics. Since BR and ME are context-free heuristics, it is possible to test their performances by comparing the correct order of a given pair of vertices to the order obtained by the *me*- and *bc*-values.

Let n denote the number of nodes in the fixed level. Then the set of edges incident with a vertex on the other level can be represented as a bit sequence of length n. The ith bit in the sequence is 1 if and only if the vertex in question is incident with the ith vertex in the fixed level. It follows that there are $2^n - 1$

Table 1 Inconsistence rates (IR) and total errors (TE) for $n=3,4,\ldots,10$

n	*BC*		*ME*		*SM*		*AM*	
	IR	*TE*	*IR*	*TE*	*IR*	*TE*	*IR*	*TE*
3	0	0	23.81	6	0	0	4.76	2
4	0	0	22.86	33	3.81	4	9.52	17
5	0.43	2	22.37	166	5.38	30	11.83	101
6	1.44	32	21.61	804	7.73	213	13.47	525
7	1.97	180	22.67	3 783	8.95	1 140	14.86	2 661
8	2.44	945	22.83	17 463	10.04	5 877	16.04	13 060
9	2.79	4 530	22.92	79 424	10.80	28 672	17.04	63 124
10	3.02	20 590	22.96	357 107	11.42	136 882	17.88	300 702

different vertices and $(2^n-1)(2^{n-1}-1)$ different pairs if vertices to be ordered. (We ignore isolated vertices and pairs of identical vertices.)

For small values of n, we can check all the vertex pairs. As mentioned already in the previous chapter we allow the heuristics to determine any order for vertices u and v for which the order does not make difference, i.e. $c_{uv}=c_{vu}$. When checking the heuristics, it soon becomes obvious that the ME heuristics often makes wrong decisions because of a phenomenon to be described below. Let $n=4$ and let the bit sequences corresponding to vertices u and v be 1010 and 1101, respectively. Thus, we have $me(u)=3$ and $me(v)=2$ resulting in an erroneous ordering. Vertex v has two edges incident with it and the ME heuristic picks up the position of the one on the right to be the value $me(v)$. This suggests that when a vertex has an even number of edges incident with it, it might be advantageous to use some other heuristic than ME.

We have tested two hybrid heuristics which differ from the ME heuristic only when a vertex has an even number of edges incident with it. If u has an even number of edges, then the *average median* heuristic (AM) assigns $am(u)$ to be the arithmetic mean of the positions of the two middle 1's on the row corresponding to u in the neighborhood matrix. (We do not consider isolated vertices.) The *semi-median* heuristic (SM) sets $sm(u)=bc(u)$ if u has an even number of edges, and $sm(u)=me(u)$ if u has an odd number of edges incident with it.

Notice that Theorem 1 holds true for the SM heuristic, while we can prove that there is a bound for the AM heuristic.

Table 1 shows the behaviors of heuristics for some small values of n. The table gives the inconsistence rates and the total errors, which are obtained by summing up the numbers of extra edge crossings caused by incorrectly ordered pairs of vertices or by unjustified ties between vertices.

The BC heuristic clearly outperforms the other heuristics when n is small. Also the hybrid heuristics AM and SM perform better than the ME heuristic. The inconsistence rates of three heuristics seem to grow rather uniformly. The ME heuristic is an interesting exception since its inconsistence rate is about the same for all values of n from three to ten. It is also worth noticing that the BC heuristic has the smallest average error size (total error/number of wrong orderings). One might suppose that Theorem 1 would imply just the opposite.

Table 2 Inconsistence rates (IR) and total errors (TE) for $n=50$ and $n=100$

n	*BC*		*ME*		*SM*		*AM*	
	IR	*TE*	*IR*	*TE*	*IR*	*TE*	*IR*	*TE*
50								
Low density	2.6	37	24.4	3596	13.2	1608	35.2	8011
Mixed density	1.6	37	19.8	2536	11.6	1361	36.0	8712
High density	1.4	40	22.0	3211	14.4	1771	36.6	9580
100								
Low density	3.2	155	14.4	4304	12.2	3167	32.8	20359
Mixed density	2.2	86	20.8	7582	13.6	3946	34.2	22888
High density	0.6	18	19.0	8238	12.4	5177	34.6	27178

We shall next study whether the results of Table 1 are valid also when n becomes larger. Since we are not any more able to check all the pairs of vertices, we use the following probabilistic process. Given the number n of the vertices in the fixed level, we randomly choose the numbers of edges incident with the vertices to be studied. We randomly choose the positions of the end points of the edges from the interval $[1,n]$, and then compare the different heuristics against the correct ordering as above. This process is repeated a fixed number of times (500 times in our tests) for values of $n=50$ and 100. We also test graphs of different density. In the low density case we choose the numbers of the edges from the interval $[0.3*n, 0.5*n]$ and in the high density case from the interval $[0.7*n, 0.9*n]$. In the mixed density case we choose one number from each interval. The results are shown in Table 2.

Again, BC is clearly the best heuristic. Contrary to Table 1, the AM heuristic is now the worst one. The ME and SM heuristics have somewhat similar performances, SM being better. As in the case of Table 1, the BC heuristic has the smallest average error size. Our tests show that the construction of the proof of Theorem 1 and similar cases fooling the BR heuristic are likely to appear very seldom in practice.

An ideal heuristic for LPP should have an inconsistence rate similar to that of BC's and a proved bound for the maximal error as ME has. The hybrid heuristics we have tested do not have the desired properties. Up to the discovery of such ideal heuristic we recommend the BR heuristic for all time critical applications. In the next chapter we consider a context-sensitive heuristic which may find drawings with fewer crossings than BC at the price of a greater time demand.

4. ON CONTEXT-SENSITIVE HEURISTICS

Eades and Kelly [4] have compared four LPP heuristics. In their test they found a context-sensitive heuristic called *splitting* to be slightly better than BC. The difference between the two heuristics was found especially when testing low density graphs.

The splitting heuristic works as follows. Arbitrarily choose a vertex u (called the

Table 3 Switches and edge crossings for $m, n = 20$, 30, and 40

n, m	*Greedy switching*		*Greedy switching with preprocessing*	
	Switches	*Edge crossings*	*Switches*	*Edge crossings*
20	74 642	102 424	15 941	71 429
30	122 280	501 296	34 065	373 594
40	174 844	1 548 842	56 878	1 210 808

pivot) and divide the vertex set into two subsets $L(u) = \{v | c_{uv} \leqq c_{vu}\}$ and $H(u) = \{v | c_{uv} > c_{vu}\}$. The heuristic is then recursively applied to $L(u)$ and $H(u)$ until a linear order is found. The number of recursive calls and hence, the time needed to apply the splitting heuristic, depends on the choice of pivots. In the worst case we need $m-1$ calls to handle a set of m vertices. In any case, the splitting heuristic always takes much more time than the BR heuristic. This obvious remark is in contrast with the results of [4, fig. 5], where the precomputations needed by context-sensitive heuristics are presumably not taken into consideration.

The following hypothesis concerning the splitting heuristic is natural: The number of recursive calls can be reduced and the solutions obtained by the heuristic can be improved by precomputing an order of the vertices according to which the pivots are chosen. The BC heuristic is an obvious choice for the precomputation. However, our tests did not support the hypothesis. The advantages of the hybride heuristic might become evident when drawing really large graphs, i.e. graphs with more than one hundred vertices per level.

The *greedy switching* [4] is a context-sensitive heuristic which starts with an arbitrary ordering and switches consecutive vertices u and v if $c_{uv} > c_{vu}$. This process continues until all consecutive vertices are in the right order. According to Eades and Kelly [4] the greedy switching does not perform quite as good as the BC or the splitting heuristics.

Again, we test whether the heuristic can be improved by preprocessing the input graph. We concentrate on low density graphs since the difference between the heuristics is small when the graphs are dense [4]. We arbitrarily choose the numbers of vertices to be at most 0.3*n, where n is the number of vertices in the fixed level.

We now face a problem tackled already in [4]. Namely, we are not able to count the inconsistence rates because of the intractability of LPP. In [4] the rates were approximated by using a certain lower bound of the total number of edges in a bipartite graph. We content ourselves with the plain numbers of edge crossings in the output graphs obtained by the greedy switching and by the greedy switching with preprocessing. Table 3 shows the results of our experiments where we arbitrarily choose 100 graphs with 20, 30, and 40 vertices in both levels.

In this case the BC preprocessing indeed improves the context-sensitive heuristic, both by decreasing the number of switches needed and by sharpening the solution. The improvement becomes relatively smaller as the graph becomes larger. In addition to the switches shown in Table 3 the heuristics make several

comparisons between vertices which are in the right order. Notice that we could consider this hybrid method as a fine tuning of the BC heuristic rather than greedy switching with preprocessing.

We end this chapter with a remark concerning the complexity of determining the values c_{uv}, i.e. the numbers of edge crossings caused by the edges incident with u and v. A direct summation formula is given in [11]. However, this formula leads to a $\Omega(n^2)$ time algorithm, where n is the number of vertices in the fixed level. Instead, we can use an auxiliary array of length n, whose ith item is defined as $A[i] = \sum_{j=1,\ldots,i-1} T[v,j]$, where T is the neighborhood matrix and the name of v is used as its number. It now follows that $c_{uv} = \sum_{i=1,\ldots,n} A[i]*T[u,i]$ (again, we indexed T by using the name of a vertex). This gives us a linear time algorithm for determining c_{uv}.

5. FINAL REMARKS

We have seen that the BC heuristic is the most preferable among the known heuristics for the level permutation problem (LPP). It can be used alone or as a preprocessing phase for some other heuristics.

Acknowledgement

This work was supported by the Academy of Finland.

References

[1] C. Batini, E. Nardelli and R. Tamassia, A layout algorithm for data-flow diagrams, *IEEE Trans. Softw. Eng.* **SE-12** (1986), 538–546.

[2] C. Batini, M. Talamo and R. Tamassia, Computer aided layout of entity-relationship diagrams, *J. Syst. Softw.* **4** (1984), 163–173.

[3] M. H. Brown, Exploring algorithms using BALSA-II, *Computer* **21** (1988), 14–36.

[4] P. Eades and D. Kelly, Heuristics for drawing 2-layered networks. *Ars Combinatoria* **21-A** (1986), 89–98.

[5] P. Eades, B. McKay and N. Wormald, An NP-complete crossing number problem for bipartite graphs. Technical Report No. 60, Dept. of Computer Science, University of Queensland, St. Lucia, April 1985.

[6] P. Eades and N. Wormald, The median heuristic for drawing 2-layered networks. Technical Report No. 69, Dept. of Computer Science, University of Queensland, St. Lucia, May 1986.

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

[8] M. R. Garey and D. S. Johnson, Crossing number is NP-complete. *SIAM J. Algebraic Discrete Methods* **4** (1983), 312–316.

[9] E. Helttula, A. Hyrskykari and K.-J. Räihä, Graphical specification of algorithm animations with Aladdin. Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences, January 1989, Kailua-Kona, Hawaii, 892–901.

[10] L. A. Rowe, M. Davis, E. Messinger, C. Meyer, C. Spirakis and A. Tuan, A browser for directed graphs. *Softw. Pract. Exper.* **17** (1987), 61–76.

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

[12] K. Sugiyama and M. Toda, Structuring information for understanding complex systems: a basis for decision making, *Fujitsu Scientific and Technical Journal* **21** (1985), 144–164.

[13] J. N. Warfield, Crossing theory and hierarchy mapping, *IEEE Trans. Syst. Man Cybern.* **SMC-7** (1977), 505–523.