Towards shortest longest edges in orthogonal graph drawing

Martin Gronemann, Michael Jünger, Sven Mallach, and Daniel R. Schmidt

Institut für Informatik, Universität zu Köln, Germany

Abstract. Inspired by a challenge during Graph Drawing 2010 'Find an orthogonal drawing whose longest edge is as short as possible', we investigate techniques to incorporate this goal into the 'standard' topology-shape-metrics approach at moderate extra computational complexity. Experiments indicate that this project is worth pursuing.

1 Introduction

In an *orthogonal drawing*, all nodes lie on a grid and all edge segments are parallel to the axes. We assume familiarity with the 'standard' basic technique to produce such drawings, the *topology-shape-metrics* (TSM), that has been introduced in the 1980s by Batini, Di Battista, Nardelli and Tamassia [3, 2]. Its topology step yields a planar(ized) embedding, the shape step an *orthogonal representation* (OR) and the metric step edge lengths from a compaction procedure.

Optimization goals include the minimization of the area of the drawing, the number of its bends and the total and maximum edge lengths. A bend-minimal OR of a four-planar embedding can be computed in time $\mathcal{O}(n^{\frac{7}{4}}\ log\ n)$ [11]. However, even given a fixed OR, the exact solution of the other optimization problems is NP-hard [16]. Even worse, finding a globally optimal solution cannot be reduced to the compaction step, but relies on all the three TSM steps. To the best of our knowledge, currently no such integrated algorithm exists.

Orthogonal compaction is strongly related to VLSI layout problems and drawings of relational diagrams. Hence, research has mainly been devoted to improvements of the area and total edge length. Concerning total edge length, Klau and Mutzel [14] provided an ILP-based exact compaction algorithm. It runs in polynomial time for restrictive kinds of ORs which are called complete or uniquely completable. Experimental studies [13, 8] show that heuristics based on turn-regularity [5] and network flows [7] typically perform best for practical and hard instances. There also exist heuristics that try to refine an already given orthogonal drawing like, e.g., [12] and [10]. Focusing on relational diagrams, in [9, 6] heuristics for compacting orthogonal drawings with fixed node sizes are presented, and in [4] an algorithm that tries to preserve a given initial sketch of a graph while computing a drawing with few bends.

In this paper we focus on keeping the longest edge as short as possible which has not yet received much attention in the literature. We restrict our attention to graphs whose nodes have a maximum degree of four and assume that the topology phase has already taken place. Since our ideas rely on the established flow-based compaction heuristics, Sect. 2 deals with the network flow problems addressed. In Sect. 3, we propose our new heuristic techniques to improve ORs and the metric step. Finally, we present an experimental analysis in Sect. 4.

2 Computing Minimum Maximum Flows

Given a *network*, i.e. a directed graph (V, E) with a source/sink $s, t \in V$, a *capacity function* $u : E \to \mathbb{Z}_{\geq 0} \cup \{\infty\}$ and *lower bounds* $l : E \to \mathbb{Z}_{\geq 0}$, an s-t-flow is a function $f : E \to \mathbb{R}_{\geq 0}$ with $l(e) \leq f(e) \leq u(e)$ on all edges $e \in E$ that has the *flow conservation* property, i.e. $\sum_{e=(v,u)\in E} f(e) - \sum_{e=(u,v)\in E} f(e) = 0$ for all $v \in V \setminus \{s, t\}$. We say that f is *integral* iff $f(e) \in \mathbb{Z}_{\geq 0}$ holds for all $e \in E$.

The *Minimum Cost Flow Problem* (MinCostFlow) in a network with costs $c : E \to \mathbb{Z}$ asks for an s-t-flow that minimizes the overall cost $\sum_{e\in E} c(e)f(e)$. For our implementation, we use the (non-polynomial but practically fast) network simplex algorithm [1]. There is also an algorithm by Garg and Tamassia [11] that was developed for a similar graph drawing problem.

Given an algorithm for the MinCostFlow problem, we can solve the *Integral Minimum Maximum s-t-Flow Problem with lower bounds* (MinMaxFlow): In a network (V, E, s, t, u) with $u \equiv \infty$, find an *integral* s-t-flow such that the maximum flow on any arc is minimum while the lower bounds are respected. The integrality constraint is important as a fractional solution can have a lower value than any integral MinMaxFlow. We can compute an integral MinMaxFlow by calling a MinCostFlow algorithm in a binary search on the capacities.

Algorithm 1: Computing a MinMaxFlow by binary search.

```
Input: A network $N = (V, E, s, t, \infty)$, lower bounds $l : E \to \mathbb{Z}_{\geq 0}$
Result: An optimal MinMaxFlow on $N$ given $l$
Solve MinCostFlow on $N$, $l$, $c \equiv 1$ and $u \equiv \infty$, obtain $f_M$;
Set $f_{max} = \max\{f_M(e) \mid e \in E\}$, $low = 0$ and $high = f_{max}$;
while $low < high$ do
    $mid = \lfloor (low + high)/2 \rfloor$;
    Solve MinCostFlow on $N$, $l$, $c \equiv 1$ and $u \equiv mid$, obtain $f_M$;
    if $f_M$ feasible then $high = mid$ else $low = mid + 1$
Solve MinCostFlow on $N$, $l$, $c \equiv 1$ and $u \equiv high$, obtain and return $f_M$;
```

The algorithm runs in time $\mathcal{O}(T_{mcf} \cdot \log f_{max})$ where T_{mcf} is the time needed by the MinCostFlow algorithm. In particular, a polynomial time MinCostFlow algorithm [1, Chapter 9], yields a polynomial time algorithm for the integral MinMaxFlow problem. For the correctness of Alg. 1, observe that every MinCostFlow is a feasible MinMaxFlow. Thus, the condition that f_M is feasible, is monotonic regarding u on $\{0, \ldots, f_{\max}\}$ and thus, binary search can be applied.

3 Minimizing the Longest Edge

In principle, the TSM approach is not well-suited for minimizing the total or maximum edge length of a drawing, since these properties depend on all three steps; yet, until the last step, there is no notion of edge lengths. However, it is state-of-the-art and widely used, so in this section we present heuristic strategies for the shape and metrics steps that can help to keep the longest edge length small. They can easily be incorporated into TSM frameworks.

3.1 MinMax Flow Compaction in the Metrics Step

Heuristic compaction is typically done in two steps. First, after making the OR turn-regular and dissecting all faces to rectangular shapes, a *constructive heuristic* is used to obtain a first feasible drawing. Then *improvement heuristics* are repeatedly applied until local optima are reached. As evaluated in [13], flow-based heuristics are a good choice. Here, constraint graphs that model adjacencies and visibilities are built for each compaction direction. Their dual graphs can be interpreted as networks N where each edge corresponds to exactly one edge of the constraint graph. After a MinCostFlow computation on N with $l \equiv 1$ and $u \equiv \infty$, the edge length can be set to dual edge flows which are of minimum total value. Instead, we use MinMaxFlow to find edge lengths so that the longest edge is short. This procedure is referred to by `ML` from now on. We also consider a variant `CML`, in which the length of the longest edge is reduced subject to the constraint that the total edge length be kept minimum. This requires an easy adaption of Alg. 1 in which the binary search direction depends on whether the current total flow exceeds its minimum value.

3.2 Balancing Bends in the Shape Step

The standard bend-minimization (`TB`) relies on a MinCostFlow problem in a dual network N. On those edges in N that correspond to primal edges, each flow unit corresponds to a bend in that primal edge. Minimizing the total number of bends can be counterproductive for our goal, as a better solution can sometimes be found by introducing further bends. Nevertheless, experiments [8] showed that bend-minimal ORs serve at least as a good basis for the compaction phase.

However, we believe that a concentration of bends on single edges has several drawbacks. First of all, it is not considered to be aesthetically pleasant. Even more, in many cases, edges with a lot of bends will be long, have complex routes or block a broad area. Heuristically, a more balanced distribution of bends could help to escape from local minima. Our results indicate that the shape step has a large influence on the achievable improvements of the compaction step. We evaluate this by applying Alg. 1 in two variations: `MB` minimizes the maximum number of bends on an edge as far as possible and `CMB` does the same as long as the total number of bends does not increase.

4 Results

We experimentally evaluate whether our balancing approach yields drawings with shorter longest edges. Our code is a modification of the TSM framework implementation in the *Open Graph Drawing Framework* (`OGDF`) [15].

For a general comparison of the different drawing strategies, we randomly generate instances[1] using `rudy` [17] and use realistic instances from the rome library [8]. From these instances, we first remove loops and isolated nodes, then all nodes with a degree of more than four in the order of their indices. If the graph becomes disconnected, we select the largest connected component that contains the smallest node index. Non-planar instances are planarized by the `OGDF` code in the topology step. The results are depicted in Tab. 1: On the `rudy` instances, minimizing the maximum number of bends per edge (`MB`) in the shape step yields the same solutions as the conventional total bend-minimization `TB`. However, using the constrained variant `CMB` in the shape step can lead to large improvements that are of the metric step approach. Yet, `CMB` can perform worse than the conventional `TB`+`TL`. Striving for short longest edges in the metric step (`ML`) is never worse than total edge length minimization (`TL`) and can lead to small improvements. `CML` performs roughly like `ML`. We observe the same on the rome library instances.

Finally, we analyze the influence of a graph's shape and density on the performance of the different drawing methods. To that end, we use `rudy` to generate complete grid graphs with integer edge weights distributed uniformly at random from 0 to 100[2]. The density of our instance is controlled by deleting all edges whose weight is at least $\Delta = 65, 70, \ldots, 95$. Disconnected instances are rejected. No planarization is needed on these instances. In Tab. 2(a), each entry depicts how many times a given strategy yielded the best result on one of the 100 instances. Here, none of the bend-minimization strategies is clearly superior, but for the non-quadratic instances, `CMB` is slightly better in total. This is in contrast to the performance of `MB` which is not better than the conventional `TB` method. For the metric step, `ML` consistently beats the other strategies; only for the very narrow 40×5 grid not much is gained. When `CMB` is used, `ML` has the highest impact for densities from 70% to 85%. Overall, `CMB`+`ML` seems to be a good choice here. A more detailed look at the performance (see Tab. 2(b)) reveals, however, that `CMB`+`ML` still can perform much worse than `TL`+`TB`.

5 Conclusion

The experiments show that one can – with little extra computational effort – put emphasis on short longest edges in the TSM framework. However, in most cases, concentrating on the metric step is not enough. For significant progress one has to integrate the shape step into the optimization process. In some cases, building the OR with the `CMB` strategy is a good idea; in other cases, it can lead to bad results. Thus, it is still unclear how to find the best OR, but the experiments show that if the right OR is found, one can expect drastic improvements.

[1] We call `rudy -rnd_graph i (200 / (i-1)) (4711 + i)` for $i = 175, \ldots, 200$. Each call yields an instance with i initial nodes that have average degree four.

[2] We call `rudy -grid_2D height width -random 0 100 (seed+i)` for $i = 1, \ldots, 100$ with an initial seed of 215986 (#inhabitants of Eindhoven according to Wikipedia).

	TB			CMB			MB		
i	TL	CML	ML	TL	CML	ML	TL	CML	ML
175	29	28	28	29	28	28	29	28	28
176	12	12	12	12	12	12	12	12	12
177	21	20	20	24	22	22	21	20	20
178	12	12	12	11	11	11	12	12	12
179	33	33	31	27	27	27	33	33	31
180	22	16	16	18	18	18	22	16	16
181	16	16	15	15	15	15	16	16	15
182	16	16	16	16	16	16	16	16	16
183	21	21	15	18	18	15	21	21	15
184	26	26	26	27	27	27	26	26	26
185	20	20	19	20	20	19	20	20	19
186	8	8	8	8	8	8	8	8	8
187	19	19	13	24	24	24	19	19	13
188	23	23	23	24	24	24	23	23	23
189	48	49	49	49	49	49	48	49	49
190	28	28	28	32	32	32	28	28	28
191	17	17	17	21	21	21	17	17	17
192	28	28	28	24	24	24	28	28	28
193	16	16	16	6	6	6	16	16	16
194	39	39	39	30	26	26	39	39	39
195	10	10	10	17	17	17	10	10	10
196	34	34	34	30	30	30	34	34	34
197	29	29	29	25	25	25	29	29	29
198	64	64	64	35	30	30	64	64	64
199	4	4	4	4	4	4	4	4	4
200	27	27	27	24	24	24	27	27	27

(a) Random `rudy` graphs.

	TB			CMB			MB		
	TL	CML	ML	TL	CML	ML	TL	CML	ML
ug10.10	6	6	6	6	6	6	6	6	6
ug20.10	2	2	2	2	2	2	2	2	2
ug30.10	8	8	8	4	4	4	8	8	8
ug40.10	4	4	4	2	2	2	4	4	4
ug50.10	4	4	4	2	2	2	4	4	4
ug10.30	12	12	11	7	7	7	12	12	11
ug20.30	7	7	7	7	7	7	7	7	7
ug30.30	12	12	12	12	12	12	12	12	12
ug40.30	10	10	10	8	8	8	10	10	10
ug50.30	6	6	6	4	4	4	6	6	6
ug10.50	18	18	18	13	11	12	18	18	18
ug20.50	10	10	10	10	10	10	10	10	10
ug30.50	20	20	20	15	15	15	20	20	20
ug40.50	10	10	10	9	9	9	10	10	10
ug50.50	20	20	20	10	10	10	20	20	20
ug20.70	16	16	16	17	17	17	16	16	16
ug30.70	38	38	38	38	38	38	38	38	38
ug40.70	31	31	31	33	33	33	31	31	31
ug50.70	16	16	16	17	17	17	16	16	16
ug60.70	38	38	38	24	23	23	38	38	38
ug10.90	23	23	23	23	23	23	23	23	23
ug20.90	25	25	25	28	28	28	25	25	25
ug30.90	45	45	45	45	45	45	45	45	45
ug40.90	25	25	25	21	21	21	25	25	25
ug50.90	22	22	22	19	16	16	22	22	22

(b) 10 to 90 node graphs from the rome-lib.

Table 1. Length of the longest edge in the computed drawing.

	#con.	TB			CMB			MB		
Δ	inst.	TL	CML	ML	TL	CML	ML	TL	CML	ML
	14 × 14 grid, seed 215986									
65	21	10	13	15	6	8	12	10	13	15
70	46	14	14	25	14	18	31	14	14	25
75	74	24	29	46	18	25	45	24	29	46
80	87	31	35	57	28	35	54	31	35	57
85	95	46	55	67	42	49	66	46	55	67
90	96	63	66	70	66	66	73	63	66	70
95	100	71	74	79	77	79	81	71	74	79
	20 × 10 grid, seed 216188									
65	14	3	4	8	1	4	8	3	4	8
70	35	11	16	22	15	16	23	11	16	22
75	66	21	26	37	28	33	45	21	26	37
80	83	30	38	51	28	41	59	30	38	51
85	96	45	51	65	54	61	74	45	51	65
90	98	61	65	71	70	73	77	61	65	71
95	100	71	73	76	61	65	70	71	73	76
	40 × 5 grid, seed 216289									
65	8	6	6	6	6	6	7	6	6	6
70	27	20	21	22	23	24	24	20	21	22
75	57	39	39	40	50	52	53	39	39	40
80	79	62	63	66	59	60	62	62	63	66
85	93	70	71	76	75	76	78	70	71	76
90	99	61	65	69	71	71	75	61	65	69
95	100	73	75	76	70	73	75	73	75	76

(a) Number of times that a given strategy yields best results. Larger values are better. The third column gives the total number of connected instances (out of 100). For $\Delta \leq 60$, only very few instances are connected.

	TB			CMB			MB		
i	TL	CML	ML	TL	CML	ML	TL	CML	ML
4	26	26	23	18	18	14	26	26	23
6	30	30	30	18	14	14	30	30	30
8	17	17	17	23	23	23	17	17	17
12	28	28	28	34	31	31	28	28	28
14	25	25	13	32	32	32	25	25	13
17	23	23	23	23	23	23	23	23	23
20	31	31	31	31	31	31	31	31	31
21	24	24	13	28	28	28	24	24	13
23	19	19	19	19	19	10	19	19	19
24	27	27	27	27	27	27	27	27	27
25	17	17	11	17	17	11	17	17	11
26	34	34	34	30	30	29	34	34	34
27	16	16	10	37	37	37	16	16	10
29	25	25	23	20	20	12	25	25	23
31	21	21	21	11	11	11	21	21	21
32	17	17	17	16	16	8	17	17	17
33	31	31	31	13	13	13	31	31	31
34	33	33	33	12	12	12	33	33	33
38	38	38	38	24	24	16	38	38	38
39	17	13	13	16	16	9	17	13	13
42	13	13	13	17	12	12	13	13	13
43	25	25	16	14	13	13	25	25	16
44	34	34	34	32	32	32	34	34	34
45	9	9	9	9	9	9	9	9	9

(b) Length of the longest edge. Smaller values are better. First 24 connected instances from the 14x14 grid, $\Delta = 70$. The new strategies work especially well on this set.

Table 2. Results on grid graphs of different shapes and densities.

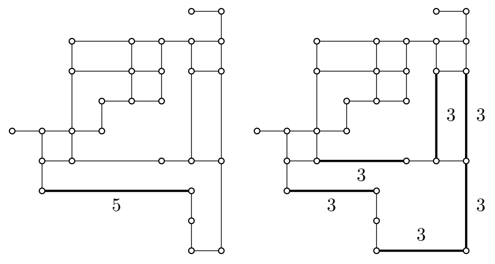

Fig. 1. Example graph compacted using `TB` with `TL` (left) and `ML` (right).

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall (1993)
2. Batini, C., Di Battista, G., Tamassia, R.: Automatic graph drawing and readability of diagrams. IEEE Trans. Syst. Man Cybern. 18, 61–79 (1988)
3. Batini, C., Nardelli, E., Tamassia, R.: A layout algorithm for data flow diagrams. IEEE Trans. Softw. Eng. 12, 538–546 (1986)
4. Brandes, U., Eiglsperger, M., Kaufmann, M., Wagner, D.: Sketch-driven orthogonal graph drawing. In: Proc. GD 2002. pp. 1–11. LNCS, Springer (2002)
5. Bridgeman, S.S., Di Battista, G., Didimo, W., Liotta, G., Tamassia, R., Vismara, L.: Turn-regularity and optimal area drawings of orthogonal representations. Comput. Geom. Theory Appl. 16, 53–93 (2000)
6. Di Battista, G., Didimo, W., Patrignani, M., Pizzonia, M.: Orthogonal and quasi-upward drawings with vertices of prescribed size. In: Proc. GD 1999. pp. 297–310. LNCS, Springer (1999)
7. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall (1999)
8. Di Battista, G., Garg, A., Liotta, G., Tamassia, R., Tassinari, E., Vargiu, F.: An experimental comparison of four graph drawing algorithms. Comput. Geom. Theory Appl. 7, 303–325 (1997)
9. Eiglsperger, M., Kaufmann, M.: Fast compaction for orthogonal drawings with vertices of prescribed size. In: Proc. GD 2001. pp. 124–138. LNCS, Springer (2002)
10. Fößmeier, U., Heß, C., Kaufmann, M.: On improving orthogonal drawings: The 4m-algorithm. In: Proc. GD 1998. pp. 125–137. LNCS, Springer (1998)
11. Garg, A., Tamassia, R.: A new minimum cost flow algorithm with applications to graph drawing. In: Proc. GD 1996. pp. 201–216. LNCS, Springer (1997)
12. Kakoulis, K.G., Six, J.M., Tollis, I.G.: Techniques for the refinement of orthogonal graph drawings. J. Graph Algorithms Appl. 4(3), 75–103 (2000)
13. Klau, G.W., Klein, K., Mutzel, P.: An experimental comparison of orthogonal compaction algorithms. In: Proc. GD 2000. pp. 37–51. LNCS, Springer (2001)
14. Klau, G.W., Mutzel, P.: Optimal compaction of orthogonal grid drawings. In: Proc. IPCO 1999. pp. 304–319. LNCS, Springer (1999)
15. OGDF: Open Graph Drawing Framework. `http://www.ogdf.net`
16. Patrignani, M.: On the complexity of orthogonal compaction. In: Proc. WADS 1999. pp. 56–61. LNCS, Springer (1999)
17. Rudy: `http://www-user.tu-chemnitz.de/~helmberg/rudy.tar.gz`