Refinement of Three-Dimensional Orthogonal Graph Drawings

Benjamin Y. S. Lynn, Antonios Symvonis, and David R. Wood

Basser Department of Computer Science
The University of Sydney
Sydney NSW 2006, Australia
{ben,symvonis,davidw}@cs.usyd.edu.au

Abstract. In this paper we introduce a number of techniques for the refinement of three-dimensional orthogonal drawings of maximum degree six graphs. We have implemented several existing algorithms for three-dimensional orthogonal graph drawing including a number of heuristics to improve their performance. The performance of the refinements on the produced drawings is then evaluated in an extensive experimental study. We measure the aesthetic criteria of the bounding box volume, the average and maximum number of bends per edge, and the average and maximum edge length. On the same set of graphs used in Di Battista *et al.* [3], our main refinement algorithm improves the above aesthetic criteria by 80%, 38%, 10%, 54% and 49%, respectively.

1 Introduction

The *3-D orthogonal grid* consists of *grid-points* in 3-space with integer coordinates, together with the axis-parallel *grid-lines* determined by these points. A *3-D orthogonal drawing* of a graph places the vertices at grid-points and routes the edges along sequences of contiguous segments of grid-lines. Edges are allowed to contain bends and can only intersect at a common vertex.

For brevity we say a 3-D orthogonal drawing of a graph G, denoted by $D(G)$, is a *drawing*. A drawing with no more than b bends per edge is called a *b-bend drawing*. The graph-theoretic terms 'vertex' and 'edge' also refer to their representation in a drawing. At a vertex v, the six directions the edges incident with v can use are called *ports*. Clearly, orthogonal drawings can only exist for graphs with maximum degree six. 3-D orthogonal graph drawings have been studied in [1,2,3,4,5,7,9,10,13,14]. By representing a vertex by a grid-box, 3-D orthogonal drawings of arbitrary degree graphs have also been considered (see [14]).

The bounding box of a given drawing is the minimum axis-parallel box which encloses the drawing. The following aesthetic criteria are the most commonly proposed measures for the quality of a given drawing.

- minimise the bounding box volume.
- minimise the maximum or average number of bends per edge.
- minimise the maximum or average length of edges.

Using straightforward extensions of the corresponding 2-D NP-hardness results, optimising any of these criteria is NP-hard [4]. In the existing algorithms for 3-D orthogonal graph drawing there is an apparent tradeoff between these aesthetic criteria, in particular, between the bounding box volume and the maximum number of bends per edge (see [5]).

Despite the fact that the drawings produced by the existing algorithms possess several desirable theoretical properties, they largely fail to communicate to the user the semantic properties of the graph being visualised. The poor visual quality of drawings produced by current algorithms can be attributed to the graph-theoretic methods which they employ. In their effort to guarantee intersection-free drawings for worst-case input graphs, they produce worst-case drawings even when the graph can be drawn in a much better way.

Post-processing refinement techniques can help rectify this situation. Here we simplify the drawings, while maintaining desired theoretic properties such as the maximum number of bends per edge route and the bounding box volume. In this paper we introduce a number of techniques for the refinement of 3-D orthogonal graph drawings. The performance of these refinements on drawings produced by several existing algorithms is then evaluated in an extensive experimental study. Refinement techniques for 2-D orthogonal drawings have been developed by Fößmeier *et al.* [6] and Six *et al.* [11].

We use the following definitions. A *direction* is an element of $\{\pm X, \pm Y, \pm Z\}$. We speak of *positive* and *negative* directions in the obvious sense. For each dimension $I \in \{X, Y, Z\}$ and direction $d = \pm I$, we say $a <_d b$, for two grid-points a and b if $I(a) < I(b)$ and d is positive, or $I(b) < I(a)$ and d is negative. A k-bend edge route vw is represented by the list $(v = b_0, b_1, b_2, \ldots, b_{k+1} = w)$, where $b_1, b_2, \ldots, b_k$ are the bends of vw. So that consecutive bends differ by at most one coordinate and there are no redundant bends, it is necessary that (1) $b_{i+1} - b_i$, $0 \leq i \leq k$, is an axis-parallel vector and (2) $b_{i+1} - b_i$ is in a different direction to $b_i - b_{i-1}$, $1 \leq i \leq k$. The length of a vector $\boldsymbol{x}$ is denoted by $|\boldsymbol{x}|$.

2 Implementation of Algorithms

This section describes the algorithms which we use to construct 3-D orthogonal drawings and particular aspects of the implementation of these algorithms which are pertinent to our experiment. For many of these algorithms, the authors were only interested in establishing asymptotic worst-case bounds for their performance, and numerous obvious improvements can be made to the algorithms, which in practice give a constant-factor improvement in some aesthetic criteria. Wherever possible, our implementations have included such improvements. For example, we remove grid-planes not containing a vertex or a bend from a given drawing, thus reducing its volume and the length of edges.

The Compact Algorithms: We now describe the Compact family of algorithms due to Eades *et al.* [5], and discuss issues relevant to their implementation. The Compact-7 algorithm positions the vertices in a $O(\sqrt{n}) \times O(\sqrt{n})$ grid in the $(Z = 0)$-plane, and produces drawings with $O(\sqrt{n}) \times O(\sqrt{n}) \times O(\sqrt{n})$ volume and at most seven bends per edge.

A critical component of the implementation of the algorithm is the construction of a graph H whose vertices correspond to the edges to be routed above the ($Z = 0$)-plane, and similarly for edges routed below the ($Z = 0$)-plane. In [5] vertices are adjacent in H if the corresponding edges start in the same row or end in the same column. A vertex-colouring of H determines the height at which edges are routed. This ensures that edges routed at the same height do not intersect. In our implementation, vertices are adjacent in H if the corresponding edge routes will intersect if routed at the same height. In general, there are less edges in H using this approach; hence in practice less colours and therefore less volume is used. We use the sequential greedy algorithm to vertex colour the graph H. Note that this method will in practice use less colours than the method of Biedl and Chan [1] which necessarily assigns a different colour to edges which start in the same row or end in the same column, even if they will not intersect if routed at the same height.

We have also implemented the Compact-6 and Compact-5 variations of the Compact algorithm, which produce drawings with at most six and five bends per edge, respectively, and with $O(\sqrt{n})\times O(\sqrt{n})\times O(n)$ and $O(\sqrt{n})\times O(n)\times O(n)$ volume, respectively. As described above for the Compact-7 algorithm we again employ an enhanced colouring method to determine the heights of edges.

Algorithms in the General Position Model: A 3-D orthogonal graph drawing is said to be in *general position* if no two vertices are in the same grid-plane. We have implemented the 3-Bends algorithm of Eades *et al.* [5] and the DLM (Diagonal Layout plus Movement) algorithm of Wood [13], both of which produce general position drawings with $O(n^3)$ volume. Drawings produced by the 3-Bends algorithm have at most three bends per edge. Drawings produced by the DLM algorithm have at most four bends per edge and an average of at most $2\frac{1}{3}$ bends per edge. For graphs with maximum degree at most five the DLM algorithm produces drawings with two bends per edge.

Ad-Hoc Algorithms: We have implemented the Incremental algorithm of Papakostas and Tollis [9] and the Reduce-Forks algorithm of Di Battista *et al.* [3]. The Incremental algorithm, which supports the on-line insertion of vertices, produces drawings with $O(n^3)$ volume and at most three bends per edge. No bounds on the volume or the number of bends have been established for the Reduce-Forks algorithm. Both of these algorithms involve a number of arbitrary decisions, thus the drawings produced may differ from one implementation to another.

Note that, due to time constraints, we have not implemented a number of algorithms in Wood [14] nor the Dynamic algorithm of Closson *et al.* [2] which produces 6-bend drawings with $O(n^2)$ volume, and supports the on-line insertion and deletion of vertices and edges.

3 Refinements

This section describes a number of techniques for the local refinement of 3-D orthogonal graph drawings. Each refinement, which can be applied to an arbitrary drawing, is aimed at improving at least one of the aesthetic criteria; namely the bounding box volume, number of bends, or the length of edges.

The MoveVertex refinement attempts to remove a bend from a given drawing by moving a vertex v to the first bend of an edge route incident to v. Applied to a drawing $D(G)$, MoveVertex(v,d) is applied for each vertex $v \in V(G)$ and direction $d \in \{\pm X, \pm Y, \pm Z\}$.

MoveVertex(vertex v, direction d)

Let $(v = a_0, a_1, a_2, \ldots)$ be the edge route, if any, using the d-port v. Let d' be the direction of the edge segment (a_1, a_2). If there is no route at v using the d' port, or if there is such an edge route $(v = b_0, b_1, b_2, \ldots)$, and $a_1 <_d b_2$, then as long as doing so does not create any new edge route intersections, move v to a_1 and reroute the edges incident to v as illustrated in Fig. 1.

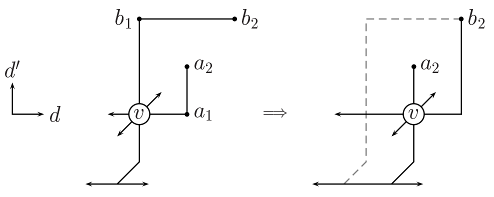


Fig. 1. The MoveVertex refinement.

The PermutePorts refinement attempts to remove bends from a given drawing by reassigning the ports at a given vertex to its incident edges. Applied to a drawing $D(G)$, PermutePorts is applied to each vertex $v \in V(G)$.

PermutePorts (vertex v)

Let $vw_1, vw_2, \ldots, vw_d$ be the edge routes incident to v, where $d = \deg(v)$. Split each edge route vw_i $1 \le i \le d$, into components vx_i and x_iw_i, where vx_i is the maximal subroute entirely contained in a grid-plane also containing v, as in Fig. 2(c). Let $S = \{vx_i : 1 \le i \le d\}$. Add to S any 0- or 1-bend edge route from v to x_i, $1 \le i \le d$, which does not intersect the remainder of the graph, as in Fig. 2(b). Find d pairwise non-intersecting edge routes in S, one for each vx_i, with the minimum total number of bends. (Such edge routes must exist since the original edge routes are in S.) Concatenate each of these edge routes with the appropriate x_iw, as in Fig. 2(c).

The RemoveSegment refinement aims to remove a bend by removing a given segment in an edge route. Applied to a drawing $D(G)$, RemoveSegment is applied for each edge $vw \in E(G)$ and pair of parallel segments (b_i, b_{i+1}) and (b_j, b_{j+1}) in vw.

RemoveSegment(edge vw, segment (b_i, b_{i+1}), segment (b_j, b_{j+1}))
Input Conditions: (b_i, b_{i+1}) and (b_j, b_{j+1}) are parallel segments of $vw = (b_0, b_1, \ldots, b_{k+1})$ such that $j > i$.

Let $\boldsymbol{x}$ be the vector $b_{i+1} - b_i$. Consider the edge route $(b_0, b_1, \ldots, b_i, b_{i+2} -$

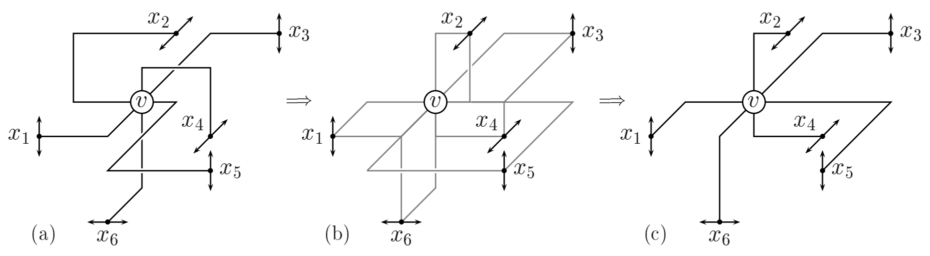


Fig. 2. Example of the PERMUTEPORTS refinement.

$\boldsymbol{x}, b_{i+3} - \boldsymbol{x}, \ldots, b_j - \boldsymbol{x}, b_{j+1}, \ldots, b_{k+1})$; that is, b_{i+1} is removed, and all grid points from b_{i+2} to b_j are translated by $-\boldsymbol{x}$. If this edge route does not intersect the remainder of the drawing, then replace vw by this route, and remove any self-intersections and redundant bends from vw.

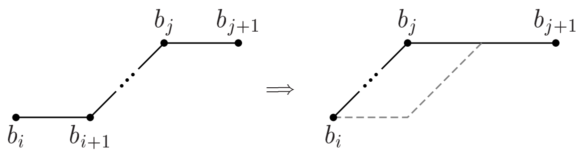


Fig. 3. The REMOVESEGMENT refinement.

The COMBINEPLANES refinement aims to reduce the volume of a given drawing by combining adjacent planes. Applied to a drawing $D(G)$, COMBINEPLANES is applied to each grid-plane in the drawing.

COMBINEPLANES(dimension I, integer x)

If the $(I = x)$-plane and the $(I = x + 1)$-plane can be combined without invalid edge or vertex intersections then do so.

The SHORTENU-TURN refinement, which is somewhat similar to the REMOVESEGMENT refinement, aims to reduce the length of a given edge by shortening parallel segments in the edge route. Applied to a drawing $D(G)$, SHORTENU-TURN$(vw, (b_i, b_{i+1}), (b_j, b_{j+1}))$ is applied for each edge $vw \in E(G)$ and pair of parallel segments (b_i, b_{i+1}) and (b_j, b_{j+1}) in vw satisfying the input conditions.

SHORTENU-TURN(edge vw, segment (b_i, b_{i+1}), segment (b_j, b_{j+1}))
Input Conditions: (b_i, b_{i+1}) and (b_j, b_{j+1}) are parallel segments of $vw = (b_0, b_1, \ldots, b_{k+1})$ such that $j > i$ and the vectors $\boldsymbol{p} = b_{i+1} - b_i$ and $\boldsymbol{q} = b_j - b_{j+1}$ are in the same direction.

Consider the route $(b_0, b_1, \ldots, b_i, b_{i+2} - \boldsymbol{x}, \ldots, b_j - \boldsymbol{x}, b_{j+1}, \ldots, b_{k+1})$ (with redundant bends also removed), where $\boldsymbol{x}$ is a vector pointing in the same direction as $\boldsymbol{p}$ with length in the range $\{0, 1, \ldots, |\boldsymbol{p}| + |\boldsymbol{q}| - 1\}$, Replace the edge route

vw with the shortest of such routes (and then with the least bends) that do not intersect the remainder of the drawing. Remove any self-intersections in vw.

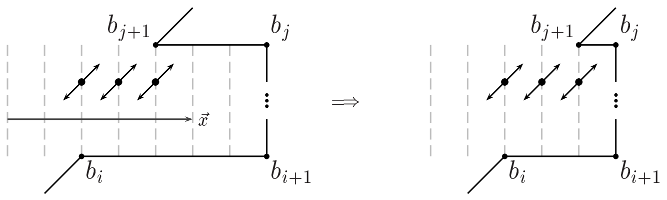


Fig. 4. An example of the ShortenU-Turn refinement.

Our final refinement, called DrawTrees&Chains, is different in nature to the previous refinements. It consists of two phases. In the first phase certain vertices are removed and certain paths are replaced by a single edge. In the second phase, which is designed to occur after other refinements have been applied, the removed vertices are reinserted, and the edge route for the single edges are replaced by paths (hopefully) with fewer edges. We now describe the first phase of the refinement.

RemoveTrees&Chains(drawing $D(G)$)

---

1. Repeatedly remove vertices with degree one (in the current drawing); that is, remove 'attached' trees from the graph.
2. We say a *chain* is a maximal path $(v_1v_2, v_2v_3, \ldots, v_{k-1}v_k)$ where every vertex v_i, except possibly for v_1 and v_k, has degree two. Replace each chain $(v_1v_2, v_2v_3, \ldots, v_{k-1}v_k)$ by the edge v_1v_k, as in Fig. 5(b).

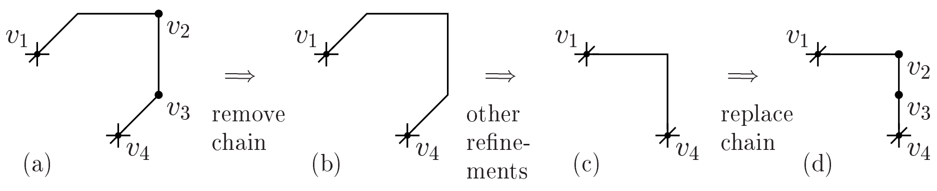
The second phase of the refinement is as follows.

RedrawTrees&Chains(drawing $D(G)$)

---

1. Insert each vertex v removed by RemoveTrees&Chains in the opposite order to their removal as follows. Let w be the adjacent vertex to v. Choose a free port at w, if any, such that vw can be routed as a unit-length segment. If such a port exists then route vw as a unit-length segment. Otherwise choose an arbitrary free port at w, insert a plane adjacent to w such that vw can be routed as a unit-length segment.
2. For each chain $(v_1v_2, v_2v_3, \ldots, v_{k-1}v_k)$ replaced by the edge route v_1v_k in the RemoveTrees&Chains phase, place the vertices $v_2, v_3, \ldots, v_{k-1}$ at the bends of the edge route v_1v_k as evenly spaced as possible. If there are more vertices than bends then position the remaining vertices arbitrarily on the edge route; see Fig. 5(d). If the edge route has less grid points than vertices then insert planes to accommodate the vertices.

We now describe how all the refinements are combined into one algorithm which we call 3D-Refine. An important decision to be made is the order



v_1
v_2
v_3
v_4
(a)
$\Longrightarrow$ remove chain
(b)
$\Longrightarrow$ other refine-ments
(c)
$\Longrightarrow$ replace chain
(d)

Fig. 5. An example of the RedrawTrees&Chains refinement.

of application of the individual refinements. To determine an optimal order, we ran five different combinations of the refinements on 36 of the *Degree-4* drawings (see Sect. 4). These drawings were determined by the Compact-7, 3-Bends, Incremental and Reduce-Forks algorithms applied to 9 graphs with $n = 15, 25, \ldots, 95$ vertices. The average percentage improvement in average bounding box side length differed by at most 4% across the five orderings of the refinements, and the average percentage improvement in average bends per edge differed by at most 5% across the five orderings of the refinements. We conclude that the ordering of the refinements is not 'significant'. The arbitrary ordering of the refinements which we chose is presented in the following algorithm.

```
3D-Refine( drawing $D(G)$)
begin
    RemoveTrees&Chains($D(G)$);
    repeat
        MoveVertex($D(G)$); PermutePorts($D(G)$);
        RemoveSegment($D(G)$); CombinePlanes($D(G)$);
        ShortenU-Turn($D(G)$);
    until no changes;
    RedrawTrees&Chains($D(G)$);
end
```

The only refinement which can possibly increase any of volume, total number of bends or total edge length is MoveVertex, which can increase the total edge length. Therefore the algorithm 3D-Refine will continue until the volume and the number of bends cannot be reduced any further, and will then only reduce the total edge length; hence the algorithm 3D-Refine will terminate.

4 The Experiment

The first set of input drawings which we use are those generated by each of the seven algorithms applied to the same graphs used in the experiment of Di Battista *et al.* [3]. These randomly generated simple connected graphs have average degree four. The authors argue that in practical graph drawing applications it is unusual to have graphs with greater density than average degree four. We call these graphs and the set of drawings produced by the algorithms applied to these graphs the *Degree-4* graphs and drawings. There are 20 graphs with n vertices for each $n = 5, 6, \ldots, 100$. Hence there are 1920 graphs and 13440 drawings.

The second set of input drawings which we use consist of randomly generated simple connected graphs which are 'almost' 6-regular. Note that this is the first experiment measuring the performance of 3-D orthogonal graph drawing algorithms on high degree graphs. The generator used here continues to add random edges to the graph until there are no pairs of non-adjacent vertices both with degree less than six. Again we have 20 graphs for each $n = 5, 6, \ldots, 100$ vertices. We call these graphs and the set of drawings produces by the algorithms applied to these graphs the *6-Regular* graphs and drawings. One would expect that 6-regular graphs are the most difficult to draw, as all ports must be used.

Performance of Individual Refinements: We now report the effects of each refinement on the input drawings. For each of the *Degree-4* and *6-Regular* drawings, we repeatedly executed each individual refinement until it cannot be applied to any portion of the drawing. This process is guaranteed to terminate since each refinement, when applied, reduces the volume of the drawing. Table 1 reports the percentage improvement of the aesthetic criteria under each individual refinement and under the 3D-REFINE algorithm, averaged over the *Degree-4* drawings and over the *6-Regular* drawings.

Table 1. Average improvements for each refinement.

Refinement	*Degree-4 Drawings*					*6-Regular Drawings*				
	Vol.	Avg. Bends	Max. Bends	Avg. Edge Len.	Max. Edge Len.	Vol.	Avg. Bends	Max. Bends	Avg. Edge Len.	Max. Edge Len.
MOVEVERTEX	35	18	0.6	20	15	5.1	1.1	0.0	2.3	2.0
PERMUTEPORTS	4.1	4.6	1.0	1.3	1.1	1.5	3.6	0.6	0.5	0.5
REMOVESEGMENT	20	18	7.3	11	9.3	11	13	6.2	6.9	6.0
COMBINEPLANES	64	4.4	0.7	31	32	50	0.9	0.3	23	24
SHORTENU-TURN	36	11	2.0	20	20	30	6.7	1.8	18	18
3D-REFINE	**80**	**38**	**10**	**53**	**49**	**57**	**15**	**7.0**	**32**	**32**

The results in Table 1 show that the refinements are considerably more effective when applied to the *Degree-4* drawings than to the *6-Regular* drawings. This is expected since the *Degree-4* graphs have average degree four, whereas the *6-Regular* graphs have high degree, and therefore the free ports at each vertex allow the refinements to be applied more often. Perhaps a more surprising observation is that the COMBINEPLANES refinement is the most successful of the refinements in terms of reducing the volume and the length of edges.

Performance of Combined Refinements: We now report the improvement in the aesthetic criteria gained by applying the 3D-REFINE algorithm to the input drawings. We firstly consider the number of bends. For most of the algorithms the number of bends (both average and maximum) per edge is consistent across all values of n — for the REDUCE-FORKS algorithm we found that the number of bends gradually increases with n. We therefore describe our results in the following manner. Table 2 reports: (1) the average and maximum number of bends per edge in drawings produced by each algorithm (averaged over the *Degree-4* graphs and the *6-Regular* graphs); (2) the average and maximum number of bends per edge in drawings obtained after applying the 3D-Refine algorithm to the drawings produced by each algorithm (averaged over the *Degree-4* graphs and the *6-Regular* graphs); (3) the percentage improvement in the average and maximum number of bends per edge gained by applying the 3D-Refine algorithm to drawings produced by each algorithm (averaged over the *Degree-4* graphs and the *6-Regular* graphs). Note that (3) is not simply the percentage improvement in (2) from (1).

Table 2. Improvements in the number of bends before and after 3D-Refine.

	Degree-4 Graphs		*6-Regular Graphs*	
	Avg. Bends	Max. Bends	Avg. Bends	Max. Bends
Compact-7	5.0 → 2.4 (52 %)	7.0 → 5.7 (19 %)	4.8 → 3.3 (32 %)	7.0 → 6.5 (7.0 %)
Compact-6	4.3 → 2.3 (46 %)	6.0 → 5.4 (10 %)	4.2 → 3.3 (22 %)	6.0 → 5.9 (2.0 %)
Compact-5	3.6 → 2.3 (36 %)	5.0 → 4.9 (1.3 %)	3.5 → 3.1 (9.8 %)	5.0 → 5.0 (0.2 %)
3-Bends	2.7 → 1.8 (31 %)	3.0 → 3.0 (0.2 %)	2.6 → 2.6 (1.2 %)	3.0 → 3.0 (0.1 %)
DLM	2.1 → 1.5 (25 %)	3.0 → 3.0 (1.0 %)	2.2 → 2.1 (4.0 %)	3.7 → 3.7 (0.0 %)
Incremental	2.1 → 1.6 (27 %)	3.0 → 3.0 (0.4 %)	2.2 → 2.1 (3.1 %)	3.0 → 3.0 (0.0 %)
Reduce-Forks	4.2 → 2.0 (48 %)	10 → 5.7 (41 %)	4.8 → 3.1 (33 %)	11 → 6.3 (37 %)
Average	3.4 → 2.0 (**38** %)	5.3 → 4.4 (**10** %)	3.5 → 2.8 (**15** %)	5.5 → 4.8 (**7.0** %)

The results in Table 2 show that the 3D-Refine algorithm considerably reduces the average number of bends in all of the *Degree-4* drawings and to a lesser extent in the *6-Regular* drawings. There is a significant improvement in the maximum number of bends per edge only in the drawings with relatively many bends prior to the application of the refinement algorithm.

Fig. 6 and Fig. 7 presents, for each algorithm, the average side length of the bounding box and the average edge length before and after the application of the 3D-Refine algorithm (averaged over the *Degree-4* and the *6-Regular* graphs, for each value of n). We chart the data for the average side length of the bounding box rather than the bounding box volume for ease of presentation — of course the average side length is the cube root of the volume. The results for the maximum edge length have been omitted; these closely resemble the analogous results for the average edge length. For each algorithm, Fig. 6 and Fig. 7 also show (next to the algorithm's name) the average percentage improvement over all graphs. This provides a measure of the susceptibility of the drawings produced by that algorithm to improvement by the 3D-Refine algorithm.

As was the case with the number of bends, the results in Fig. 6 and Fig. 7 show that the 3D-Refine algorithm considerably reduces the bounding box volume and the length of edges in all of the *Degree-4* drawings and to a lesser

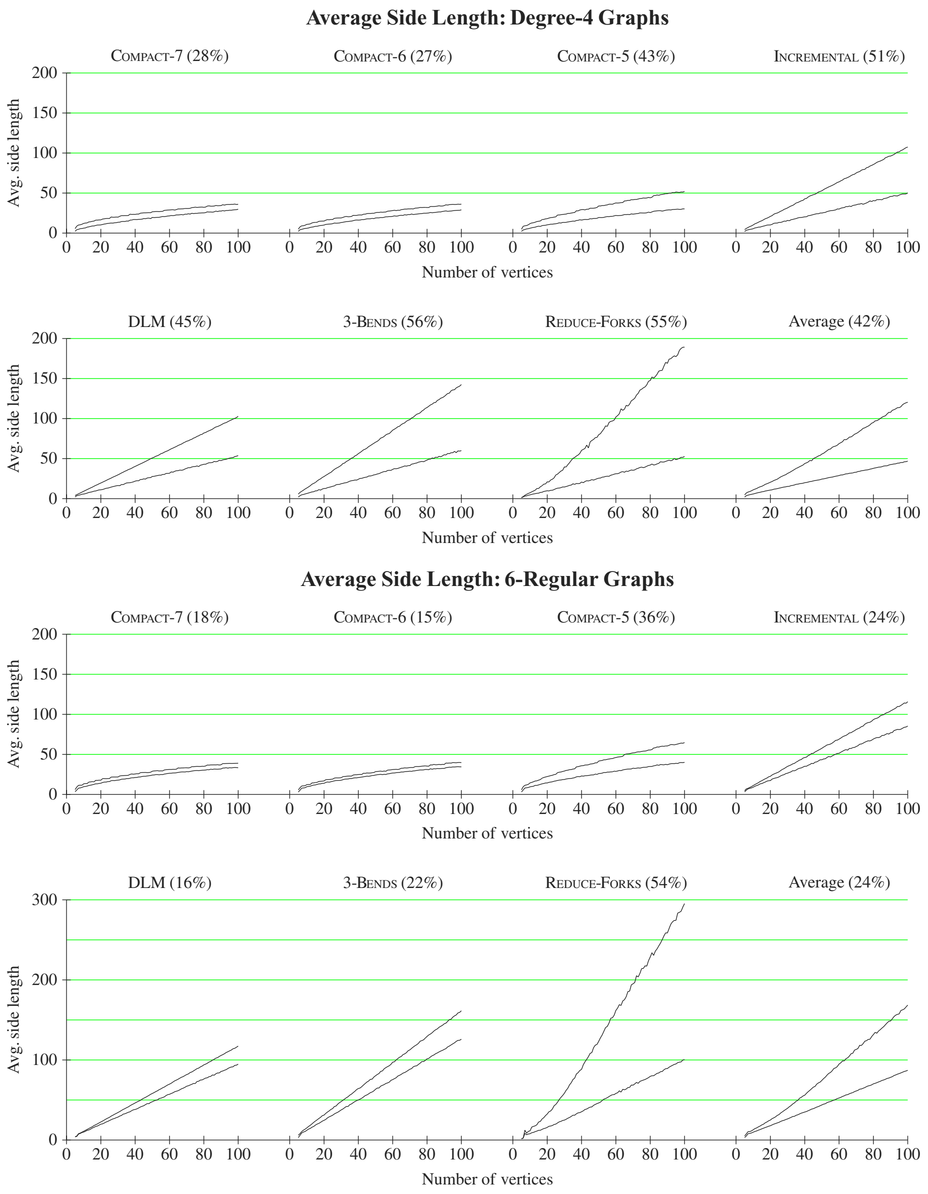


Fig. 6. Average bounding box side length before and after 3D-Refine.

extent in the *6-Regular* drawings. Note that the average improvement of 42% for the average bounding box side length obtained for the *Degree-4* drawings corresponds to an improvement of 80% in the bounding box volume.

Comparison of Algorithms: We now compare the performance of the drawing algorithms, firstly without refinements and then following the application of the 3D-Refine algorithm; see Table 2, Fig. 6 and Fig. 7. Our results generally

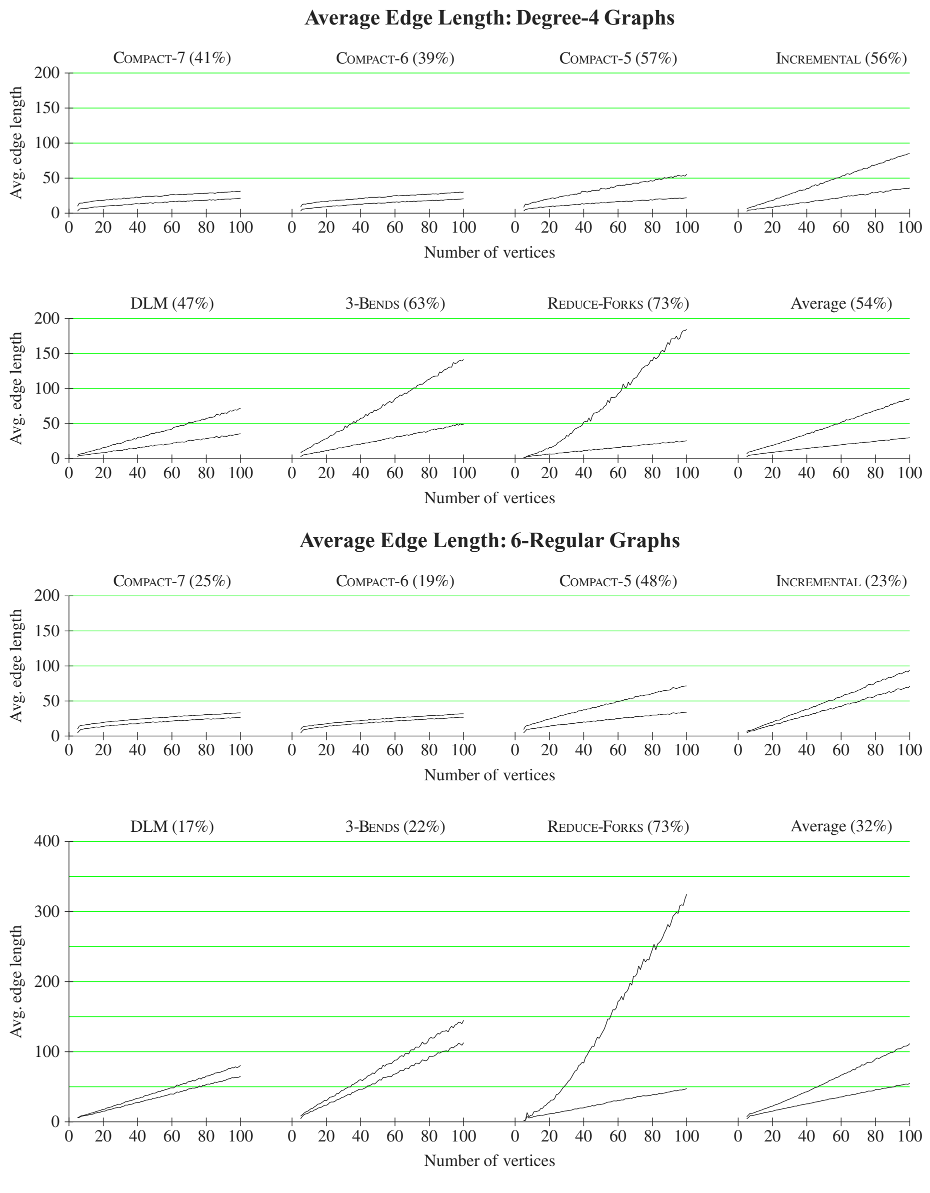


Fig. 7. Average edge length before and after 3D-Refine.

confirm the worst case upper bounds established for each algorithm, and confirm the results in [3] for the *Degree-4* graphs. The performance of the algorithms on the *6-Regular* graphs continue the trends observed for the *Degree-4* graphs with one marked exception. The Reduce-Forks algorithm performs noticeably worse on the *6-Regular* graphs relative to the other algorithms compared with the *Degree-4* graphs.

In terms of volume and edge lengths the best algorithms (without refinements) are Compact7, Compact6 and Compact5. Note that the worst-case volume bounds for Compact7 and Compact6 are $O(n^{3/2})$ and $O(n^2)$, respectively. That in practice the difference in their performance is negligible is due to the enhanced colouring method discussed in Sect. 2. DLM and Incremental are the next best performing algorithms, followed by 3-Bends and Reduce-Forks. A similar pattern emerges when comparing the algorithms after refinements, except that Reduce-Forks performs almost as well as the Compact algorithms; that is, the drawings produced by the Reduce-Forks algorithm are highly susceptible to improvements by 3D-Refine.

5 Conclusion

In this paper we have described a number of post-processing techniques for the refinement of 3-D orthogonal graph drawings. Our main algorithm makes substantial improvements to all of the aesthetic criteria measured, especially when applied to relatively low degree graphs. This experiment has contributed to the ongoing research efforts to make 3-D orthogonal graph drawings more appropriate for visualisation purposes. An important future step toward this goal is the development of efficient data structures for the implementation of algorithms and refinements for 3-D orthogonal graph drawing.

Acknowledgements. Thanks to Chi Nguyen for technical support, and to Maurizio Patrignani for kindly supplying the code from the 3DCube implementations of the Incremental and Reduce-Forks algorithms.

References

1. T. Biedl and T. Chan. Cross-coloring: improving the technique by Kolmogorov and Barzdin. Technical Report CS-2000-13, University of Waterloo, Canada, 2000.
2. M. Closson, S. Gartshore, J. Johansen, and S. K. Wismath. Fully dynamic 3-dimensional orthogonal graph drawing. In Kratochvil [8], pages 49–58.
3. G. Di Battista, M. Patrignani, and F. Vargiu. A split&push approach to 3D orthogonal drawing. In Whitesides [12], pages 87–101.
4. P. Eades, C. Stirk, and S. Whitesides. The techniques of Komolgorov and Bardzin for three dimensional orthogonal graph drawings. *Inform. Proc. Lett.*, 60(2):97–103, 1996.
5. P. Eades, A. Symvonis, and S. Whitesides. Three dimensional orthogonal graph drawing algorithms. *Discrete Applied Math.*, 103:55–87, 2000.
6. U. Fößmeier, C. Heß, and M. Kaufmann. On improving orthogonal drawings: the $4M$-algorithm. In Whitesides [12], pages 125–137.
7. A. N. Kolmogorov and Ya. M. Barzdin. On the realization of nets in 3-dimensional space. *Problems in Cybernetics*, 8:261–268, March 1967.
8. J. Kratochvil, editor. *Proc. Graph Drawing: 7th International Symp. (GD'99)*, volume 1731 of *Lecture Notes in Comput. Sci.*, Springer, 1999.
9. A. Papakostas and I. G. Tollis. Algorithms for incremental orthogonal graph drawing in three dimensions. *J. Graph Algorithms Appl.*, 3(4):81–115, 1999.

10. M. Patrignani and F. Vargiu. 3DCube: a tool for three dimensional graph drawing. In G. Di Battista, editor, *Proc. Graph Drawing: 5th International Symp. (GD'97)*, volume 1353 of *Lecture Notes in Comput. Sci.*, pages 284–290, Springer, 1998.
11. J. M. Six, K. G. Kakoulis, and I. G. Tollis. Refinement of orthogonal graph drawings. In Whitesides [12], pages 302–315.
12. S. Whitesides, editor. *Proc. Graph Drawing: 6th International Symp. (GD'98)*, volume 1547 of *Lecture Notes in Comput. Sci.*, Springer, 1998.
13. D. R. Wood. An algorithm for three-dimensional orthogonal graph drawing. In Whitesides [12], pages 332–346.
14. D. R. Wood. *Three-Dimensional Orthogonal Graph Drawing.* PhD thesis, School of Computer Science and Software Engineering, Monash University, Australia, 2000.