Optimal Compaction of Orthogonal Grid Drawings

Gunnar W. Klau   Petra Mutzel

MPI–I–98–1–031   December 1998

Authors' Addresses

Gunnar W. Klau
Max–Planck–Institut für Informatik
Im Stadtwald
66123 Saarbrücken
guwek@mpi-sb.mpg.de
http://www.mpi-sb.mpg.de/~guwek

Petra Mutzel
Max–Planck–Institut für Informatik
Im Stadtwald
66123 Saarbrücken
mutzel@mpi-sb.mpg.de
http://www.mpi-sb.mpg.de/~mutzel

Publication Notes

This work is partially supported by the Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (No. 03–MU7MP1–4).

Abstract

We consider the two–dimensional compaction problem for orthogonal grid drawings in which the task is to alter the coordinates of the vertices and edge segments while preserving the shape of the drawing so that the total edge length is minimized. The problem is closely related to two–dimensional compaction in VLSI–design and is conjectured to be *NP*–hard.

We characterize the set of feasible solutions for the two–dimensional compaction problem in terms of paths in the so–called constraint graphs in x– and y–direction. Similar graphs (known as *layout graphs*) have already been used for one–dimensional compaction in VLSI–design, but this is the first time that a direct connection between these graphs is established. Given the pair of constraint graphs, the two–dimensional compaction task can be viewed as extending these graphs by new arcs so that certain conditions are satisfied and the total edge length is minimized. We can recognize those instances having only one such extension; for these cases we can solve the compaction problem in polynomial time.

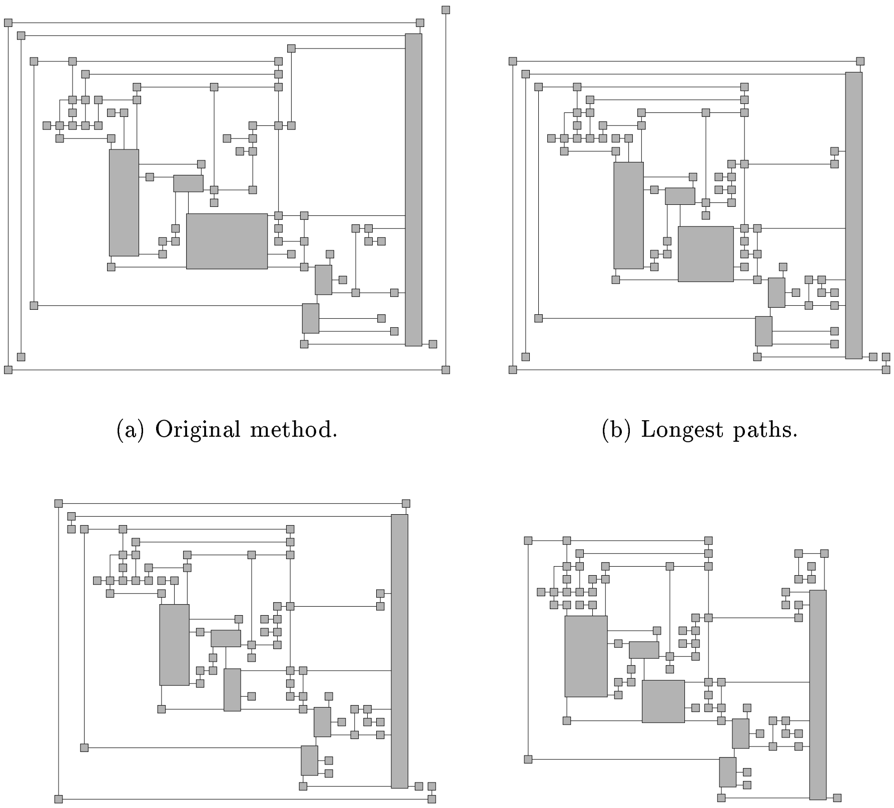
We have transformed the geometrical problem into a graph–theoretical one which can be formulated as an integer linear program. Our computational experiments have shown that the new approach works well in practice. It is the first time that the two–dimensional compaction problem is formulated as an integer linear program.

1 Introduction

The compaction problem has been one of the challenging tasks in VLSI–design. The goal is to minimize the area or total edge length of the circuit layout while mainly preserving its shape. In graph drawing the compaction problem also plays an important role. Orthogonal grid drawings, in particular the ones produced by the algorithms based on bend minimization (e.g., [Tam87, FK96, KM98]), suffer from missing compaction algorithms. In orthogonal grid drawings every edge is represented as a chain of horizontal and vertical segments; moreover, the vertices and bends are placed on grid points. Two orthogonal drawings have the same shape if one can be obtained from the other by modifying only the lengths of the horizontal and vertical segments without changing the angles formed by them. The orthogonal drawing standard is in particular suitable for Entity–Relationship diagrams, state charts, and PERT–diagrams with applications in data bases, CASE–tools, algorithm engineering, work–flow management and many more.

So far, in graph drawing only heuristics have been used for compacting orthogonal grid drawings. Tamassia [Tam87] suggested "refining" the shape of the drawing into one with rectangular faces by introducing artificial edges. If all the faces are rectangles, the compaction problem can be solved in polynomial time using minimum–cost network flow algorithms. In general, however, the solution is far from the optimal solution for the original graph without the artificial edges (see also Section 4). Other heuristics are based on the idea of iteratively fixing the x– and y–coordinates followed by a one–dimensional compaction step. In one–dimensional compaction, the goal is to minimize the width or height of the layout while preserving the coordinates of the fixed dimension. It can be solved in polynomial time using the so–called *layout graphs*. The differences of the methods are illustrated in Fig. 1. It shows the output of four different compaction methods for a given graph with fixed shape. The big vertices are internally represented as a face that must have rectangular shape. Figure 1(a) illustrates the method proposed in [Tam87]. Figures 1(b) and 1(c) show the output of two one–dimensional compaction strategies applied to the drawing in Fig. 1(a). Both methods use the underlying layout graphs, the first algorithm is based on longest paths computations, the second on computing minimum–cost flows. Figure 1(d) shows an optimally compacted drawing computed by our algorithm.

The compaction problems in VLSI–design and in graph drawing are closely related but are not the same. Most versions of the compaction problem in VLSI–design are proven to be *NP*–hard [Len90]. For this reason, the compaction problem in graph drawing is also conjectured to be *NP*–hard. Research on the compaction problem in VLSI–design has concentrated on one–dimensional methods. Only two papers have suggested optimal algorithms for two–dimensional compaction (see [SLW83, KW84]). The idea is based on a $(0, 1)$–non–linear integer programming formulation of the prob-



(a) Original method.

(b) Longest paths.

(c) Flow.

(d) Optimal.

Figure 1: The result of four different compaction algorithms.

lem and solving it via branch–and–bound. Unfortunately, the methods have not been efficient enough for practical use.

Lengauer states the following: *"The difficulty of two–dimensional compaction lies in determining how the two dimensions of the layout must interact to minimize the area"* [Len90, p.581]. The two–dimensional compaction method presented in this paper exactly attacks this point. We provide a necessary and sufficient condition for all feasible solutions of a given instance of the compaction problem. This condition is based on existing paths in the so–called *constraint graphs* in x– and y–direction. These graphs are similar to the *layout graphs* known from one–dimensional compaction methods. The layout graphs, however, are based on visibility properties whereas the constraint graphs arise from the shape of the drawing. As far as we know, this is the first time that direct connections between the two graphs have been used for two–dimensional compaction.

Let us describe our idea more precisely: The two constraint graphs D_h and D_v specify the shape of the given orthogonal drawing. We characterize exactly those extensions of the constraint graphs which belong to feasible orthogonal grid drawings. The task is to extend the given constraint graphs to a complete pair of constraint graphs by adding new arcs for which the necessary and sufficient condition is satisfied and the total edge–length of the layout is minimized. Hence, we have transformed the geometrical problem into a graph–theoretical one. Furthermore, we can detect constructively those instances having only one complete extension. For these cases, we can solve the compaction problem in polynomial time.

We formulate the resulting problem as an integer linear program that can be solved via branch–and–cut or branch–and–bound methods. Our computational results on a benchmark set of 11,582 graphs [BGL+97] have shown that we are able to solve the two–dimensional compaction problem for all the instances in short computation time. Furthermore, they have shown that often it is worthwhile looking for the optimally compacted drawing. The total edge lengths have been improved up to $37,0\%$ and $65,4\%$ as compared to iterated one–dimensional compaction and the method proposed in [Tam87].

In Section 2 we provide the characterization of the set of feasible solutions. The formulation of the compaction problem in the form of an integer linear program is given in Section 3. We describe a realization of our approach and corresponding computational experiments on a set of benchmark graphs in Section 4. Finally, Section 5 contains the conclusions and our future plans.

2 A Characterization of Feasible Solutions

In this section we describe the transformation of the compaction problem into a graph–theoretical problem. After giving some basic definitions we introduce the notion of shape descriptions and present some of their properties. As a main result of this section, we establish a one–to–one correspondence between shape descriptions satisfying a certain property and feasible orthogonal grid drawings.

2.1 Definitions and Notations

In an *orthogonal (grid) drawing* Γ of a graph G the vertices are placed on mutually distinct integer grid points and the edges on mutually line–distinct paths in the grid connecting their endpoints. We call Γ *simple* if the number of bends and crossings in Γ is zero. Each orthogonal grid drawing can be transformed into a simple orthogonal grid drawing. The straightforward transformation consists of replacing each crossing and bend by a virtual vertex. Figure 2 gives an example.

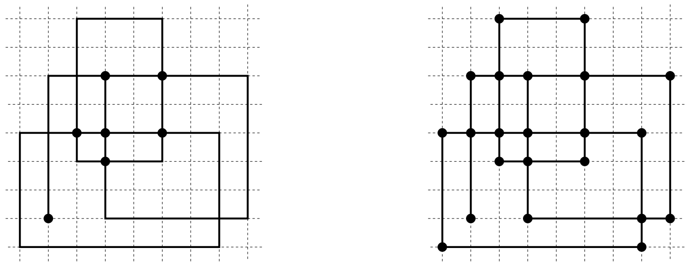


Figure 2: An orthogonal grid drawing and its simple counterpart.

The shape of a simple drawing is given by the angles inside the faces, i.e., the angles occurring at consecutive edges of a face cycle and the angles formed by bends on the edges. Note that the notion of shape induces a partitioning of drawings in equivalence classes. Often, the shape of an orthogonal drawing is given by a so–called *orthogonal representation* H. Formally, for a simple orthogonal drawing, H is a function from the set of faces F to clockwise ordered lists of tuples (e_r, a_r) where e_r is an edge, and a_r is the angle formed with the following edge inside the appropriate face. Using the definitions, we can specify the compaction problem:

Definition 1. The *compaction problem for orthogonal drawings* (COD) is stated by the following: Given a simple orthogonal grid drawing Γ with orthogonal representation H, find a drawing Γ' with orthogonal representation H of minimum total edge length.

The compaction problem (COD) is conjectured to be *NP*–hard. So far, in practice, one–dimensional graph–based compaction methods have been used. But the results are often not satisfying. Figure 3(a) shows an orthogonal drawing with total edge length $2k + 5$ which cannot be improved by one–dimensional compaction methods. The reason for this lies in the fact that one–dimensional compaction methods are based on visibility properties. An exact two–dimensional compaction method computes Fig. 3(b) in which the total edge length is only $k + 6$.

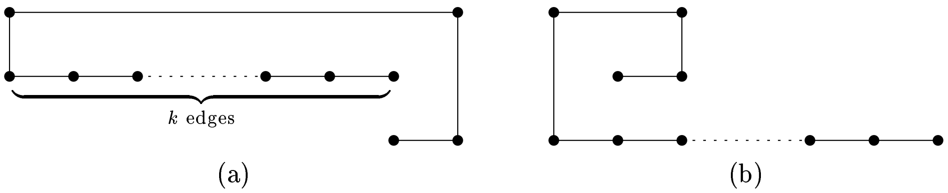


Figure 3: A drawing generated with (a) an iterative one–dimensional and (b) an optimal compaction method.

Let Γ be a simple orthogonal grid drawing of a graph $G = (V, E)$. It induces a partition of the set of edges E into the horizontal set E_h and the vertical set E_v. A horizontal (resp. vertical) *subsegment* in Γ is a connected component in (V, E_h) (resp. (V, E_v)). If the component is maximally connected it is also referred to as a *segment*. We denote the set of horizontal and vertical subsegments by $\mathcal{S}_h$ and $\mathcal{S}_v$ resp., and the sets of segments by $S_h \subseteq \mathcal{S}_h$, $S_v \subseteq \mathcal{S}_v$. We further define $\mathcal{S} = \mathcal{S}_h \cup \mathcal{S}_v$ and $S = S_h \cup S_v$. The following observations are of interest (see also Fig. 4).

1. Each edge is a subsegment, i.e., $E_h \subseteq \mathcal{S}_h, E_v \subseteq \mathcal{S}_v$.

2. Each vertex belongs to one horizontal and one vertical segment, denoted by $\text{hor}(v)$ and $\text{vert}(v)$.

3. Each subsegment s is contained in exactly one segment, referred to as $\text{seg}(s)$.

4. The *limits* of a subsegment s are given as follows: If s is horizontal, let v and w be its leftmost and rightmost vertices, if s is vertical let v and w be the bottommost and topmost vertices on s. The horizontal limits of s are denoted as $\alpha_h(s)$ and $\Omega_h(s)$ and specify the segments to the left and to the right of s. Equivalently, we define the vertical limits $\alpha_v(s)$ and $\Omega_v(s)$. The following table contains the precise definitions.

direction of s	$\alpha_h(s)$	$\Omega_h(s)$	$\alpha_v(s)$	$\Omega_v(s)$
horizontal	$\text{vert}(v)$	$\text{vert}(w)$	$\text{seg}(s)$	$\text{seg}(s)$
vertical	$\text{seg}(s)$	$\text{seg}(s)$	$\text{hor}(v)$	$\text{hor}(w)$

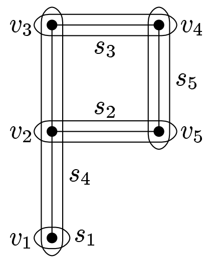
The following lemma implies that the total number of segments is $2|V| - |E|$.

Lemma 1. *Let Γ be a simple orthogonal grid drawing of a graph $G = (V, E_h \cup E_v)$. Then $|S_h| = |V| - |E_h|$ and $|S_v| = |V| - |E_v|$.*

Proof. Induction on the number of edges $m = m_h + m_v$. Induction basis ($m = 0$): The maximally connected components consist of single vertices. Thus, $|S_h| = |S_v| = |V|$. Induction step ($m \to m' = m + 1$): Without loss of generality, let the additional edge be horizontal, causing a union of two horizontal segments. Then $|V'| = |V|$, $m'_h = m_h + 1$ and $m'_v = m_v$. It follows, that $|S'_v| = |S_v| = |V| - m_v = |V'| - m'_v$ and $|S'_h| = |S_h| - 1 = |V| - m_h - 1 = |V'| - m'_h$. □

2.2 Shape Descriptions and its Properties

Let $G = (V, E)$ be a graph with simple orthogonal representation H and segments $S_h \cup S_v$. A *shape description* of H is a tuple $\sigma = \langle D_h, D_v \rangle$ of



$S_h = \{s_1, s_2, s_3\}$ $\quad$ $S_v = \{s_4, s_5\}$

i	$V(s_i)$	$E(s_i)$
1	$\{v_1\}$	$\{\}$
2	$\{v_2, v_5\}$	$\{(v_2, v_5)\}$
3	$\{v_3, v_4\}$	$\{(v_3, v_4)\}$
4	$\{v_1, v_2, v_3\}$	$\{(v_1, v_2), (v_2, v_3)\}$
5	$\{v_4, v_5\}$	$\{(v_4, v_5)\}$

i	$\alpha_h(s_i)$	$\Omega_h(s_i)$	$\alpha_v(s_i)$	$\Omega_v(s_i)$
1	s_4	s_4	s_1	s_1
2	s_4	s_5	s_2	s_2
3	s_4	s_5	s_3	s_3
4	s_4	s_4	s_1	s_3
5	s_5	s_5	s_2	s_3

Figure 4: Segments of a simple orthogonal grid drawing and its limits.

so–called *constraint graphs*. Both graphs are directed and defined as $D_h = (S_v, A_h)$ and $D_v = (S_h, A_v)$. Thus, each node in D_h and D_v is a segment and the arc sets are given by

$$A_h = \{(\alpha_h(e), \Omega_h(e)) \mid e \in E_h\} \quad \text{and} \quad A_v = \{(\alpha_v(e), \Omega_v(e)) \mid e \in E_v\}.$$

The two digraphs characterize known relationships between the segments that must hold in any drawing of the graph because of its shape properties. Let $a = (s_i, s_j)$ be an arc in $A_h \cup A_v$. If $a \in A_v$ then the horizontal segment s_i must be placed at least one grid unit below segment s_j. For vertical segments the arc $a \in A_h$ expresses the fact that s_i must be to the left of s_j. Each arc is defined by at least one edge in E. Clearly, each vertical edge determines the relative position of two horizontal segments and vice versa. Figure 5 illustrates the shape description of a simple orthogonal grid drawing.

For two vertices v and w we use the notation $v \xrightarrow{*} w$ if there is a path from v to w. Shape descriptions have the following property:

Lemma 2. *Let $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ be a shape description. For every subsegment $s \in \mathcal{S}_h \cup \mathcal{S}_v$ the paths $\alpha_h(s) \xrightarrow{*} \Omega_h(s)$ and $\alpha_v(s) \xrightarrow{*} \Omega_v(s)$ are contained in $A_h \cup A_v$.*

Proof. We prove the lemma for horizontal subsegments. The proof for vertical subsegments is similar. By definition, the lemma holds for edges. Let s be a horizontal subsegment consisting of k consecutive edges $e_1, \ldots, e_k$. With $\alpha_h(e_1) = \alpha_h(s)$, $\Omega_h(e_k) = \Omega_h(s)$ and $(\alpha_h(e_i), \Omega_h(e_i)) \in A_h \cup A_v$ and $\alpha_v(s) = \Omega_v(s) = \mathrm{seg}(s)$ the result follows. $\square$

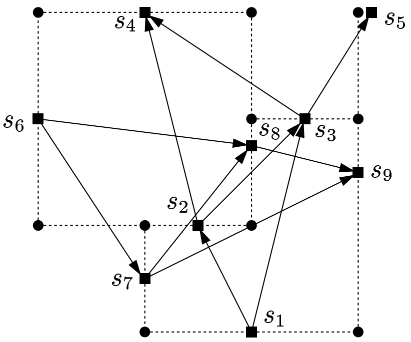


Figure 5: A simple orthogonal drawing (dotted) and its shape description. $S_h = \{s_1, \dots, s_5\}$ and $S_v = \{s_6, \dots, s_9\}$.

Definition 2. Let $(s_i, s_j) \in \mathcal{S} \times \mathcal{S}$ be a pair of subsegments. We call the pair *separated* iff one of the following conditions holds:

1. $\Omega_h(s_i) \xrightarrow{*} \alpha_h(s_j)$ (s_i is to the left of s_j)
2. $\Omega_h(s_j) \xrightarrow{*} \alpha_h(s_i)$ (s_i is to the right of s_j)
3. $\Omega_v(s_j) \xrightarrow{*} \alpha_v(s_i)$ (s_i is above s_j)
4. $\Omega_v(s_i) \xrightarrow{*} \alpha_v(s_j)$ (s_i is below s_j)

In an orthogonal drawing at least one of the four conditions must be satisfied for any such pair (s_i, s_j) where s_i and s_j are not adjacent. The following two observations show that we only need to consider separated segments of opposite direction.

Observation 1. *Let $(s_i, s_j) \in \mathcal{S} \times \mathcal{S}$ be a non–adjacent pair of subsegments. If $(seg(s_i), seg(s_j))$ is separated then (s_i, s_j) is separated.*

Proof. Note that $\alpha_h(\text{seg}(s_i)) \xrightarrow{*} \alpha_h(s_i) \xrightarrow{*} \Omega_h(s_i) \xrightarrow{*} \Omega_h(\text{seg}(s_i))$ and $\alpha_h(\text{seg}(s_j)) \xrightarrow{*} \alpha_h(s_j) \xrightarrow{*} \Omega_h(s_j) \xrightarrow{*} \Omega_h(\text{seg}(s_j))$ according to Lemma 2. The same holds for the corresponding vertical paths. Let seg(s_i) and seg(s_j) be separated. Combined with Definition 2, this yields the separation of s_i and s_j. □

Observation 2. *Assume that the arcs between the segments form an acyclic graph. Then the following is true: All non–adjacent segment pairs are separated if and only if all non–adjacent segment pairs of opposite directions are separated.*

Proof. The forward direction of the proof is trivial. For the backward direction, assume that there is a non–adjacent pair $(s_i, s_j) \in S \times S$ which is not separated and every non–adjacent pair of opposite directions is separated. Consider the following cases:

1. $(s_i, s_j) \in S_h \times S_v$ or $(s_i, s_j) \in S_v \times S_h$. This is a contradiction to the assumption.

2. $(s_i, s_j) \in S_h \times S_h$. Let $L = \{\alpha_h(s_i), \Omega_h(s_i), \alpha_h(s_j), \Omega_h(s_j)\}$ be the horizontal limits of s_i and s_j. For each vertical segment $l \in L$, we define $h(l)$ as the segment which is limited by l and $p(l)$ as the other horizontal segment. More precisely:

l	$h(l)$	$p(l)$
$\alpha_h(s_i)$	s_i	s_j
$\Omega_h(s_i)$	s_i	s_j
$\alpha_h(s_j)$	s_j	s_i
$\Omega_h(s_j)$	s_j	s_i

Now consider the four obviously non–adjacent pairs of opposite direction in $\bigcup_{l \in L} \{(l, p(l))\}$. Note that $\alpha_v(h(l)) = h(l) = \Omega_v(h(l))$ and $\alpha_v(l) \xrightarrow{*} h(l) \xrightarrow{*} \Omega_v(l)$ for all $l \in L$. It follows that none of the pairs is vertically separated, since

$$\begin{array}{lrcl} & \Omega_v(p(l)) \xrightarrow{*} \alpha_v(l) & \vee & \Omega_v(l) \xrightarrow{*} \alpha_v(p(l)) \\ \Leftrightarrow & p(l) \xrightarrow{*} \alpha_v(l) & \vee & \Omega_v(l) \xrightarrow{*} p(l) \\ \Rightarrow & p(l) \xrightarrow{*} h(l) & \vee & h(l) \xrightarrow{*} p(l) \\ \Leftrightarrow & s_i \xrightarrow{*} s_j & \vee & s_j \xrightarrow{*} s_i, \end{array}$$

which is a contradiction to the separation of s_i and s_j.

Concerning the horizontal separation, note that $\alpha(l) = \Omega_h(l) = l$ for all $l \in L$. The horizontal part of Definition 2 for every $l \in L$ reads as

$$\begin{array}{lrcl} & \Omega_h(l) \xrightarrow{*} \alpha_h(p(l)) & \vee & \Omega_h(p(l)) \xrightarrow{*} \alpha_h(l) \\ \Leftrightarrow & l \xrightarrow{*} \alpha_h(p(l)) & \vee & \Omega_h(p(l)) \xrightarrow{*} l. \end{array}$$

For each l, one of the two terms implies the separation of s_i and s_j and can be eliminated. The remaining four terms induce two cycles in the graph, which is again a contradiction to the assumption.

3. $(s_i, s_j) \in S_v \times S_v$. This case is similar to the previous one. $\square$

The following lemma shows that we can restrict our focus to separated segments that share a common face. For a face f, we write $S(f)$ for the segments containing the horizontal and vertical edges on the boundary of f.

Lemma 3. *All non–adjacent segment pairs are separated if and only if for every face f the non–adjacent pairs of segments $(s_i, s_j) \in S(f) \times S(f)$ are separated.*

Proof. The forward direction of the proof follows immediately because of $S(f) \subseteq S$ for every face f. To show the other direction we consider two adjacent faces f and g, both satisfying the condition. Assume that there is a pair of segments (s_i, s_j) that is not separated, without loss of generality let $s_i \in S(f)$ and $s_j \in S(g)$ as in Fig. 6. Then there must exist a non–separated segment pair $(s'_i, s'_j) \in S(g) \times S(g)$, which is a contradiction. Since adjacency is a transitive relation, the proposition is true for all faces and thus for the whole graph. □

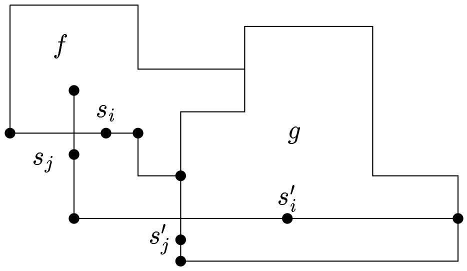


Figure 6: Two adjacent faces with non–separated segments s_i and s_j.

2.3 Complete Extensions of Shape Descriptions

We will see next that any shape description $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ can be extended so that the resulting constraint graphs correspond to a feasible orthogonal planar drawing. We give a characterization of these *complete extensions* in terms of properties of their constraint graphs.

Definition 3. A *complete extension* of a shape description $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ is a tuple $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ with the following properties:

1. $A_h \subseteq B_h, A_v \subseteq B_v$.
2. B_h and B_v are acyclic.
3. Every non–adjacent segment pair is separated.

The following theorem characterizes the set of feasible solutions for the compaction problem.

Theorem 1. *For any simple orthogonal drawing with shape description $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ there exists a complete extension $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ of σ and vice versa: Any complete extension $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ of a shape description $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ corresponds to a simple orthogonal drawing with shape description σ.*

Proof. To prove the first part of the theorem, we consider a simple orthogonal grid drawing Γ with shape description $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$. Let $c(s_i)$ denote the fixed coordinate for segment $s_i \in S_h \cup S_v$. We construct a complete extension $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ for σ as follows: $B_h = \{(s_i, s_j) \in S_v \times S_v \mid c(s_i) < c(s_j)\}$, i.e., we insert an arc from every vertical segment to each vertical segment lying to the right of s_i. Similarly, we construct the set B_v. Clearly, we have $A_h \subseteq B_h$ and $A_v \subseteq B_v$. We show the completeness by contradiction: Assume first that there is some non–adjacent pair (s_i, s_j) which is not separated. According to the construction this is only possible if the segments cross in Γ, which is a contradiction. Now assume that there is a cycle in one of the arc sets. Again, the construction of B_h and B_v forbids this case. Hence τ is a complete extension of σ.

We give a constructive proof for the second part of the theorem by specifying a simple orthogonal grid drawing for the complete extension τ. To accomplish this task we need to assign lengths to the segments. A *length assignment* for a complete extension of a shape description $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ is a function $c : S_v \cup S_h \to \mathbb{N}$ with the property $(s_i, s_j) \in B_h \cup B_v \Rightarrow c(s_i) < c(s_j)$. Given τ, such a function can be computed using any topological sorting algorithm in the acyclic graphs in τ, e.g., longest paths or minimum–cost flow algorithms in the dual graph. For a fixed length assignment, the following simple and straightforward method assigns coordinates to the vertices. Let $x \in \mathbb{N}^V$ and $y \in \mathbb{N}^V$ be the coordinate vectors. Then simply setting $x_v = c(\mathrm{vert}(v))$ and $y_v = c(\mathrm{hor}(v))$ for every vertex $v \in V$ results in a correct grid drawing. The following points have to be verified:

1. All edges have positive integer length.
   The length of an edge $e \in E$ is given by $c(\Omega(e)) - c(\alpha(e))$. We know that both values are integer and according to Lemma 2 that $(\alpha(e), \Omega(e)) \in B_h \cup B_v$ and thus $c(\Omega(e)) > c(\alpha(e))$.

2. Γ maps each circuit in G into a rectilinear polygon.
   For a face f let $\Gamma(f)$ be the geometric representation of vertices and edges belonging to f. It it sufficient to show that $\Gamma(f)$ is a rectilinear polygon for each face f in G. Every vertex v on the boundary of f is placed according to the segments $\mathrm{hor}(v)$ and $\mathrm{vert}(v)$. Two consecutive vertices v and w on the boundary of f either share the same horizontal or the same vertical segment (since they are linked by an edge). Thus, either $x_v = x_w$ or $y_v = y_w$.

3. No non–adjacent subsegments cross.
   Otherwise, assume that there are two such segments s_i and s_j that cross. Then $c(\Omega_h(s_i)) \geq c(\alpha_h(s_j))$, $c(\Omega_h(s_j)) \geq c(\alpha_h(s_i))$, $c(\Omega_v(s_j)) \geq c(\alpha_v(s_i))$, and $c(\Omega_v(s_i)) \geq c(\alpha_v(s_j))$ (see also Fig. 7). This is a contradiction to the completeness of τ. $\square$

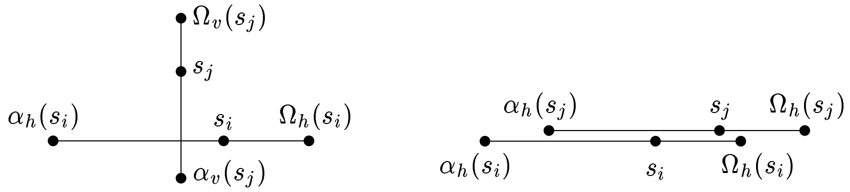


Figure 7: Crossing subsegments of opposite (left) and same (right) direction.

We have transformed the compaction problem into a graph–theoretical problem. Our new task is to find a complete extension of a given shape description σ that minimizes the total edge length. If a shape description already satisfies the conditions of a complete extension (see Fig. 8(a)), the compaction problem can be solved optimally in polynomial time: The corresponding inequalities form a totally unimodular matrix (see also the proof of Observation 3). Sometimes the shape description is not complete but it is only possible to extend it in one way (see Fig. 8(b)). In these cases also it is easy to solve the compaction problem. But in most cases it is not clear how to extend the shape description since there are many different possibilities (see Fig. 8(c)).

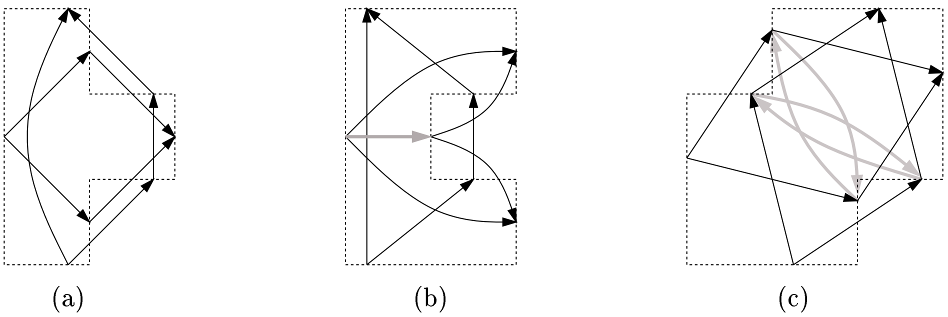


Figure 8: Three types of shape descriptions. The dotted lines show the orthogonal grid drawings, thin arrows arcs in shape descriptions and the thick gray arcs possible completions.

Observe that the preprocessing phase of our algorithm described at the beginning of Section 4 can test in polynomial time if a shape description can be uniquely completed. In that case, our algorithm does the completion and computes an optimal drawing in polynomial time.

3 An ILP Formulation for the Compaction Problem

The characterization given in the previous section can be used to obtain an integer linear programming formulation for the compaction problem COD. Let Γ be a simple orthogonal grid drawing of a graph $G = (V, E_h \cup E_v)$ with orthogonal representation H and let $\sigma = \langle (S_v, A_h), (S_h, A_v) \rangle$ be the corresponding shape description. The set of feasible solutions for this instance of COD can now be written as $\mathcal{C}(\sigma) = \{\tau \mid \tau \text{ is a complete extension of } \sigma\}$. Let $C = S_h \times S_h \cup S_v \times S_v$ be the set of possible arcs in the digraphs of σ. $\mathbb{Q}^{|C|}$ is the vector space whose elements are indexed with numbers corresponding to the members of C. For a complete extension $\tau = \langle (S_v, B_h), (S_h, B_v) \rangle$ of σ we define an element $x^\tau \in \mathbb{Q}^{|C|}$ in the following way: $x^\tau_{ij} = 1$ if $(s_i, s_j) \in B_h \cup B_v$ and $x^\tau_{ij} = 0$ otherwise. We use these vectors to characterize the *Compaction Polytope* $P_{\text{COD}} = \text{conv}\{x^\tau \in \mathbb{Q}^{|C|} \mid \tau \in \mathcal{C}(\sigma)\}$.

In order to determine the minimum total edge length over all feasible points in P_{COD} we introduce a vector $c \in \mathbb{Q}^{|S_h \cup S_v|}$ to code the length assignment and give an integer linear programming formulation for the compaction problem COD. Let M be a very large number (for the choice of M see Lemma 5). The ILP for the compaction problem is the following:

$$\text{(ILP)} \qquad \min \sum_{e \in E_h} c_{\Omega_h(e)} - c_{\alpha_h(e)} + \sum_{e \in E_v} c_{\Omega_v(e)} - c_{\alpha_v(e)} \qquad \text{subject to}$$

$$\begin{array}{rll}
x_{ij} = 1 & \forall (s_i, s_j) \in A_h \cup A_v & (1.1) \\
x_{\Omega_h(i),\alpha_h(j)} + x_{\Omega_h(j),\alpha_h(i)} + & \forall (s_i, s_j) \in S \times S, & \\
x_{\Omega_v(j),\alpha_v(i)} + x_{\Omega_v(i),\alpha_v(j)} \geq 1 & (s_i, s_j) \text{ non-adjacent} & (1.2) \\
c_i - c_j + (M+1) x_{ij} \leq M & \forall (s_i, s_j) \in C & (1.3) \\
0 \leq x_{ij} \leq 1 & \forall (s_i, s_j) \in C & (1.4) \\
c_i \geq 0 & \forall s_i \in S_h \cup S_v & (1.5) \\
x_{ij} \in \mathbb{N} & \forall (s_i, s_j) \in C & (1.6)
\end{array}$$

The objective function expresses the total edge length in a drawing for G. Note that the formulation also captures the related problem of minimizing the length of the longest edge in a drawing. In this case, the constraints $c_{\Omega_h(e)} - c_{\alpha_h(e)} \leq l_{\max}$ or $c_{\Omega_v(e)} - c_{\alpha_v(e)} \leq l_{\max}$ must be added for each edge e and the objective function must be substituted by $\min l_{\max}$. Furthermore, it is possible to give each edge an individual weight in the objective function. In this manner, edges with higher values are considered more important and will preferably be assigned a shorter length. In VLSI–design, the weight factor is usually chosen according to the electric resistivity of the corresponding wire. Here, wires with high resistivity should be short in the resulting layout.

We first give an informal motivation of the three different types of constraints and then show that any feasible solution of the ILP formulation indeed corresponds to an orthogonal grid drawing.

(1.1) *Shape constraints.* We are looking for an extension of shape description σ. Since any extension must contain the arc sets of σ, the appropriate entries of x must be set to 1.

(1.2) *Completeness constraints.* This set of constraints guarantees completeness of the extension. The respective inequalities model Definition 3.3.

(1.3) *Consistency constraints.* The vector c corresponds to the length assignment and thus must fulfill the property $(s_i, s_j) \in B_h \cup B_v \Rightarrow c(s_i) < c(s_j)$. If $x_{ij} = 1$, then the inequality reads $c_j - c_i \geq 1$, realizing the property for the arc (i, j). The case $x_{ij} = 0$ yields $c_i - c_j \leq M$ which is true if M is set to the maximum of the width and the height of Γ. The choice of M is discussed in Lemma 5.

The following observation shows that we do not need to require integrality for the variable vector c. The subsequent lemma motivates the fact that no additional constraints forbidding the cycles are necessary.

Observation 3. *Let (x, c_f) with $x \in \{0,1\}^{|C|}$ and $c_f \in \mathbb{Q}^{|S_h \cup S_v|}$ be a feasible solution for (ILP) and let z_f be the value of the objective function. Then there is also a feasible solution (x, c) with $c \in \mathbb{N}^{|S_h \cup S_v|}$ and objective function value $z \leq z_f$.*

Proof. Since x is part of a feasible solution, its components must be either zero or one. Then (ILP) reads as a totally unimodular optimization problem: Consider that the consistency constraints for $x_{ij} = 0$ can be ignored and the remaining constraints form a totally unimodular matrix and thus define an integral polyhedron [NW88]. Note that a fixed x in a solution corresponds to a complete extension of the given shape description. □

Lemma 4. *Let (x, c) be a feasible solution for ILP 1 and let D_h and D_v be the digraphs corresponding to x. Then D_h and D_v are acyclic.*

Proof. Assume without loss of generality that there is a cycle $(s_{i_1}, s_{i_2}, \ldots, s_{i_k})$ of length k in D_h. This implies $x_{i_1,i_2} = x_{i_2,i_3} = \ldots = x_{i_k,i_1} = 1$ and the appropriate consistency constraints read

$$\begin{array}{rcl} c_{i_1} - c_{i_2} + (M+1) & \leq & M \\ c_{i_2} - c_{i_3} + (M+1) & \leq & M \\ & \vdots & \\ c_{i_k} - c_{i_1} + (M+1) & \leq & M \end{array}$$

Summed up, this yields $k \cdot (M+1) \leq k \cdot M$ which is equivalent to $M + 1 \leq M$. □

Theorem 2. *For each feasible solution (x, c) of (ILP) for a shape description σ, there is a simple orthogonal grid drawing Γ whose shape corresponds to σ and vice versa. The total edge length of Γ is equal to the value of the objective function.*

Proof. For the first part of the proof, let x and c be the solution vectors of (ILP). According to Observation 3 we can assume that both vectors are integer. Vector x describes a complete extension τ of σ; the extension property is guaranteed by constraints 1.1, completeness by constraints 1.2 and acyclicity according to Lemma 4. The consistency constraints 1.3 require c to be a length assignment for τ. With Theorem 1 the result follows.

Again, we give a constructive proof for the second part: Theorem 1 allows us to use Γ for the construction of a complete extension τ for σ. Setting c_i to the fixed coordinate of segment s_i and x_{ij} to 1 if the arc (s_i, s_j) is contained in τ and to 0 otherwise, results in a feasible solution for the ILP. Evidently, the bounds and integrality requirements for c and x are not violated and constraints 1.1 and 1.2 are clearly satisfied. To show that the consistency constraints hold, we consider two arbitrary vertical segments s_i and s_j. Two cases may occur: Assume first that s_i is to the left of s_j. Then the corresponding constraint reads $c_j - c_i \geq 1$ which is true since $c(j) > c(i)$ and the values are integer. Now suppose that s_i is not to the left of s_j. In this case the constraint becomes $c_i - c_j \leq M$ which is true for a sufficiently large M. For the choice of M see Lemma 5. A similar discussion applies to horizontal segments. Obviously, the value of the objective function is equal to the total edge length in both directions of the proof. □

The (ILP) can be solved via a branch–and–cut or branch–and–bound algorithm. We now discuss the choice of the constant M.

Lemma 5. *The value* $\max\{|S_h|, |S_v|\}$ *is a sufficient choice for M.*

Proof. Note that any optimal drawing Γ has width $w \leq |S_v|$ and height $h \leq |S_h|$, otherwise a one–dimensional compaction step could be applied. M has to be big enough to "disable" the constraints 1.3 if the corresponding entry of x is zero; it has to be an upper bound for the distance between any pair of segments. Setting it to the size of the bigger of the two sets fulfills this requirement. □

4 Implementation and Computational Results

In this section we describe our implementation to solve the ILP presented in the previous section. We report on results based on a large set of benchmark graphs, comparing our implementation to standard heuristics.

4.1 Solving the ILP

Our implementation which we will refer to as OPT throughout this section, solves the integer linear program described in Section 3. It is realized as a compaction module inside the AGD–library [AGMN97, AGD98] and is written in C^{++} using LEDA [MN95, MNSU98].

In a preprocessing phase, the given shape description σ is completed as far as possible. Starting with a list L of possibly non–separated segments, we remove a pair (s_i, s_j) from L if it either fulfills the definition of separation or there is only one possibility to meet this requirement. In the latter case we insert the appropriate arc and proceed the iteration. The process stops if no such pair can be found in L. If the list is empty at the end of the preprocessing phase, we can solve the compaction problem in polynomial time by optimizing over the integral polyhedron given by the corresponding inequalities.

Otherwise, let σ^+ be the extension of σ resulting from the preprocessing phase. OPT first computes a corrupted layout for σ^+ in polynomial time. Then it checks which non–separated pairs in L induce indeed a violation of the drawing and adds the corresponding completeness and consistency inequalities. The resulting integer linear program is solved with the Mixed Integer Solver from CPLEX. The process of checking, adding inequalities and solving the ILP may have to be repeated because new pairs in L can cause a violation. However, this iteration of optimization algorithms has shown to be superior to adding all the constraints corresponding to entries in L and solving the resulting ILP. The disadvantage of this method is that we do not have any quality guarantee during the computation unless the algorithm processes the last ILP. Currently we are realizing a second implementation within the branch–and–cut framework ABACUS [JT98].

For the initial construction of L we exploit Lemma 3 and Observation 2. We only have to add pairs of segments which share a common face and which are of opposite direction. By decreasing the size of L in this way we could reduce the search space and speed up the computation remarkably.

4.2 Computational Results

We compare the results of our method to the results achieved by two other compaction methods: ORIG is an implementation of the original method proposed in [Tam87]. It divides all the faces of the drawing into sub–faces of rectangular shape and assigns consistent edge lengths in linear time. 1DIM is an optimal one–dimensional compaction algorithm. It first calls ORIG to get an initial drawing and then runs iteratively a visibility–based compaction, alternating the direction in each run. It stops if no further one–dimensional improvement is possible.

The algorithms ORIG, 1DIM, and OPT have been tested on a large test set. This set, collected by the group of G. Di Battista, contains 11,582 graphs representing data from real–world applications [BGL+97]. For our experimental study, each of the graphs has been planarized by computing a planar subgraph and reinserting the edges, representing each crossing by a virtual vertex. The distribution of the resulting graphs is shown in Fig. 9. After fixing the planar embedding for every graph we computed its shape using an extension of Tamassia's bend minimizing algorithm [KM98]. The resulting graphs with fixed shape served as input for the three compaction algorithms. We compared the total edge length and the area of the smallest enclosing rectangle of the drawings produced by ORIG, 1DIM, and OPT. Furthermore, we recorded their running times.

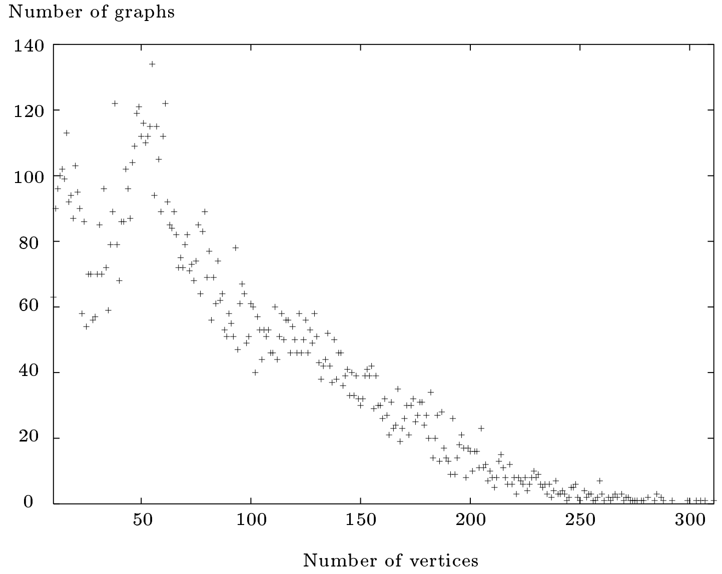


Figure 9: Distribution of graphs in the test suite.

All the examples could be solved to optimality on a SUN Enterprise 10000. For all instances, the running times of ORIG and 1DIM have been below 0.05 and 0.43 seconds, respectively. For OPT, the distribution of running times is shown in Fig. 10. The vast majority of instances could be solved in less than one second, few graphs needed more than five seconds. The longest running time was 68 minutes.

The average improvement of the total edge length computed by OPT over the whole test set of graphs was 2.4% compared to 1DIM and 21,0% compared to ORIG. Just looking at hard instances where OPT needed more than two seconds of running time we yield average improvements of 7.5% and 36.1%, resp. Figures 11 and 12 show this fact in more detail: The x–axis denotes the size of the graphs, the y–axis shows the improvement of total edge length in percent with respect to 1DIM and ORIG. We computed the minimal, maximal and average improvement for the graphs of the same size.

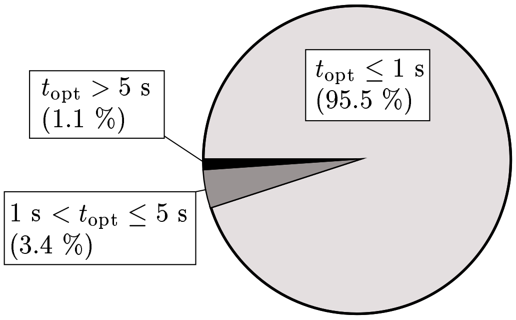


Figure 10: Running times for OPT.

The average improvement values are quite independent from the graph size, and the minimum and maximum values converge to them with increasing number of vertices. Note that OPT yields in some cases improvements of more than 30% in comparison to the previously best strategy, Fig. 1 in Section 1 shows the drawings for such a case (here, the improvement is 28%). For the area, the data look similar with the restriction that in few cases the values produced by 1DIM are slightly better than those from OPT; short edge length does not necessarily lead to low area consumption. The average area improvements compared to 1DIM and ORIG are 2.7% and 29.3%, however.

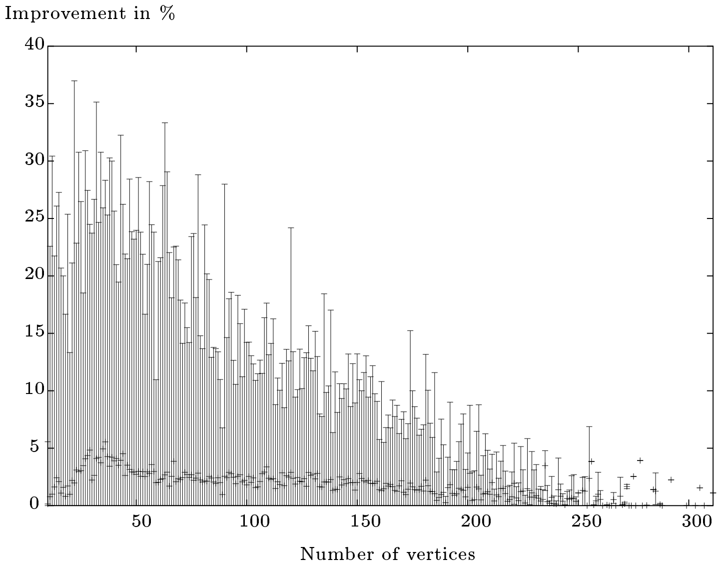


Figure 11: Quality of OPT compared to 1DIM.

In general, we could make the following observations: Instances of the compaction problem COD divide into easy and hard problems, depending on

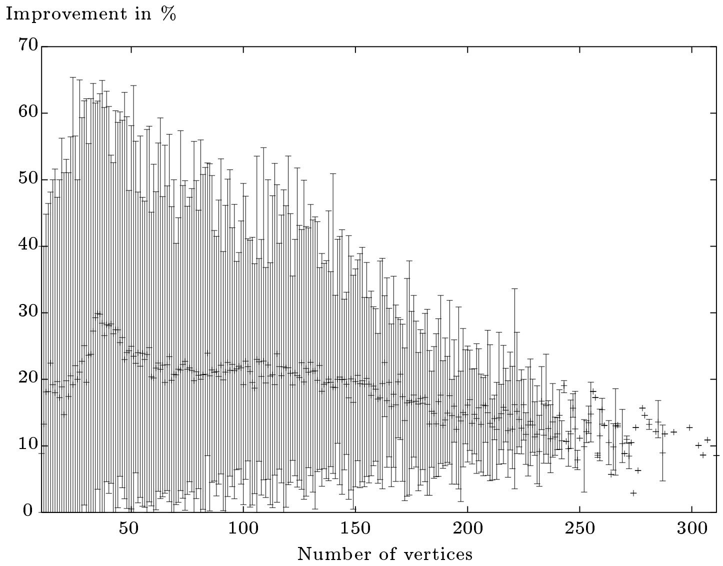


Figure 12: Quality of OPT compared to ORIG.

the structure of the corresponding graphs. On the one hand, we are able to solve some randomly generated instances of biconnected planar graphs with 1,000 vertices in less than five seconds. In these cases, however, the improvement compared to the results computed by 1DIM is small. On the other hand, graphs containing tree–like structures have shown to be hard to compact since their number of fundamentally different drawings is in general very high. For these cases, however, the improvement is much higher.

5 Conclusions

We have introduced the constraint graphs describing the shape of a simple orthogonal grid drawing. Furthermore, we have established a direct connection between these graphs by defining *complete extensions* of the constraint graphs that satisfy certain connectivity requirements in both graphs. We have shown that complete extensions characterize the set of feasible drawings with the given shape. For a given complete extension we can solve the compaction problem COD in polynomial time. The graph–theoretical characterization allows us to formulate COD as an integer linear program. The preprocessing phase of our algorithm detects those instances having only one complete extension and, for them, the optimal algorithm runs in polynomial time. Our experiments show that the resulting ILP can be solved within short computation time for instances as big as 1,000 vertices.

There are still open questions concerning the two–dimensional compaction problem. So far, there is no *NP*–hardness proof for COD. Furthermore, we are not satisfied having the 'big' M in our integer linear programming formulation. We expect further improvements on the running time of our algorithm by implementing it as a branch–and–cut algorithm.

References

[AGD98] AGD. *AGD User Manual.* Max-Planck-Institut Saarbrücken, Universität Halle, Universität Köln, 1998. Available via `http://www.mpi-sb.mpg.de/AGD/`. Partially supported by the DFG-cluster "Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen".

[AGMN97] D. Alberts, C. Gutwenger, P. Mutzel, and S. Näher. AGD–Library: A library of algorithms for graph drawing. In G.F. Italiano and S. Orlando, editors, *WAE '97 (Proc. on the Workshop on Algorithm Engineering)*, Venice, Italy, Sept. 11-13, 1997. `http://www.dsi.unive.it/~wae97`.

[BGL+97] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *CGTA: Computational Geometry: Theory and Applications*, 7:303 – 316, 1997.

[FK96] U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. In Franz J. Brandenburg, editor, *Graph Drawing (Proc. GD '95)*, volume 1027 of *Lecture Notes in Computer Science*, pages 254–266. Springer-Verlag, 1996.

[JT98] M. Jünger and S. Thienel. Introduction to ABACUS - A Branch-And-CUt System. *Operations Research Letters*, 22:83–95, March 1998.

[KM98] G. W. Klau and P. Mutzel. Quasi–orthogonal drawing of planar graphs. Technical Report MPI–I–98–1–013, Max–Planck–Institut für Informatik, Saarbrücken, May 1998.

[KW84] G. Kedem and H. Watanabe. Graph optimization techniques for IC–layout and compaction. *IEEE Transact. Comp.-Aided Design of Integrated Circuits and Systems*, CAD-3 (1):12–20, 1984.

[Len90] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* John Wiley & Sons, New York, 1990.

[MN95] K. Mehlhorn and S. Näher. LEDA: A platform for combinatorial and geometric computing. *Communications of the ACM*, 38(1):96–102, 1995.

[MNSU98] K. Mehlhorn, S. Näher, M. Seel, and C. Uhrig. LEDA Manual Version 3.7. Technical report, Max-Planck-Institut für Informatik, 1998. Available via `http://www.mpi-sb.mpg.de/LEDA`.

[NW88] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley and Sons, New York, 1988.

[SLW83] M. Schlag, Y.-Z. Liao, and C. K. Wong. An algorithm for optimal two–dimensional compaction of VLSI layouts. *Integration, the VLSI Journal*, 1:179–209, 1983.

[Tam87] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987.

INFORMATIK

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Birgit Hofmann
Im Stadtwald
D-66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-98-2-018	F. Eisenbrand	A Note on the Membership Problem for the First Elementary Closure of a Polyhedron
MPI-I-98-2-017	M. Tzakova, P. Blackburn	Hybridizing Concept Languages
MPI-I-98-2-014	Y. Gurevich, M. Veanes	Partisan Corroboration, and Shifted Pairing
MPI-I-98-2-013	H. Ganzinger, F. Jacquemard, M. Veanes	Rigid Reachability
MPI-I-98-2-012	G. Delzanno, A. Podelski	Model Checking Infinite-state Systems in CLP
MPI-I-98-2-011	A. Degtyarev, A. Voronkov	Equality Reasoning in Sequent-Based Calculi
MPI-I-98-2-010	S. Ramangalahy	Strategies for Conformance Testing
MPI-I-98-2-009	S. Vorobyov	The Undecidability of the First-Order Theories of One Step Rewriting in Linear Canonical Systems
MPI-I-98-2-008	S. Vorobyov	AE-Equational theory of context unification is Co-RE-Hard
MPI-I-98-2-007	S. Vorobyov	The Most Nonelementary Theory (A Direct Lower Bound Proof)
MPI-I-98-2-006	P. Blackburn, M. Tzakova	Hybrid Languages and Temporal Logic
MPI-I-98-2-005	M. Veanes	The Relation Between Second-Order Unification and Simultaneous Rigid E-Unification
MPI-I-98-2-004	S. Vorobyov	Satisfiability of Functional+Record Subtype Constraints is NP-Hard
MPI-I-98-2-003	R.A. Schmidt	E-Unification for Subsystems of S4
MPI-I-98-1-030	H. Brönniman, L. Kettner, S. Schirra, R. Veltkamp	Applications of the Generic Programming Paradigm in the Design of CGAL
MPI-I-98-1-029	P. Mutzel, R. Weiskircher	Optimizing Over All Combinatorial Embeddings of a Planar Graph
MPI-I-98-1-028	A. Crauser, K. Mehlhorn, E. Althaus, K. Brengel, T. Buchheit, J. Keller, H. Krone, O. Lambert, R. Schulte, S. Thiel, M. Westphal, R. Wirth	On the performance of LEDA-SM
MPI-I-98-1-027	C. Burnikel	Delaunay Graphs by Divide and Conquer
MPI-I-98-1-026	K. Jansen, L. Porkolab	Improved Approximation Schemes for Scheduling Unrelated Parallel Machines
MPI-I-98-1-025	K. Jansen, L. Porkolab	Linear-time Approximation Schemes for Scheduling Malleable Parallel Tasks
MPI-I-98-1-024	S. Burkhardt, A. Crauser, P. Ferragina, H. Lenhof, E. Rivals, M. Vingron	q-gram Based Database Searching Using a Suffix Array (QUASAR)
MPI-I-98-1-023	C. Burnikel	Rational Points on Circles

MPI-I-98-1-022	C. Burnikel, J. Ziegler	Fast Recursive Division
MPI-I-98-1-021	S. Albers, G. Schmidt	Scheduling with Unexpected Machine Breakdowns
MPI-I-98-1-020	C. Rüb	On Wallace's Method for the Generation of Normal Variates
MPI-I-98-1-019		2nd Workshop on Algorithm Engineering WAE '98 - Proceedings
MPI-I-98-1-018	D. Dubhashi, D. Ranjan	On Positive Influence and Negative Dependence
MPI-I-98-1-017	A. Crauser, P. Ferragina, K. Mehlhorn, U. Meyer, E. Ramos	Randomized External-Memory Algorithms for Some Geometric Problems
MPI-I-98-1-016	P. Krysta, K. Loryś	New Approximation Algorithms for the Achromatic Number
MPI-I-98-1-015	M.R. Henzinger, S. Leonardi	Scheduling Multicasts on Unit-Capacity Trees and Meshes
MPI-I-98-1-014	U. Meyer, J.F. Sibeyn	Time-Independent Gossiping on Full-Port Tori
MPI-I-98-1-013	G.W. Klau, P. Mutzel	Quasi-Orthogonal Drawing of Planar Graphs
MPI-I-98-1-012	S. Mahajan, E.A. Ramos, K.V. Subrahmanyam	Solving some discrepancy problems in NC*
MPI-I-98-1-011	G.N. Frederickson, R. Solis-Oba	Robustness analysis in combinatorial optimization
MPI-I-98-1-010	R. Solis-Oba	2-Approximation algorithm for finding a spanning tree with maximum number of leaves
MPI-I-98-1-009	D. Frigioni, A. Marchetti-Spaccamela, U. Nanni	Fully dynamic shortest paths and negative cycle detection on diagraphs with Arbitrary Arc Weights
MPI-I-98-1-008	M. Jünger, S. Leipert, P. Mutzel	A Note on Computing a Maximal Planar Subgraph using PQ-Trees
MPI-I-98-1-007	A. Fabri, G. Giezeman, L. Kettner, S. Schirra, S. Schönherr	On the Design of CGAL, the Computational Geometry Algorithms Library
MPI-I-98-1-006	K. Jansen	A new characterization for parity graphs and a coloring problem with costs
MPI-I-98-1-005	K. Jansen	The mutual exclusion scheduling problem for permutation and comparability graphs
MPI-I-98-1-004	S. Schirra	Robustness and Precision Issues in Geometric Computation
MPI-I-98-1-003	S. Schirra	Parameterized Implementations of Classical Planar Convex Hull Algorithms and Extreme Point Compuations
MPI-I-98-1-002	G.S. Brodal, M.C. Pinotti	Comparator Networks for Binary Heap Construction
MPI-I-98-1-001	T. Hagerup	Simpler and Faster Static AC^0 Dictionaries
MPI-I-97-2-012	L. Bachmair, H. Ganzinger, A. Voronkov	Elimination of Equality via Transformation with Ordering Constraints
MPI-I-97-2-011	L. Bachmair, H. Ganzinger	Strict Basic Superposition and Chaining
MPI-I-97-2-010	S. Vorobyov, A. Voronkov	Complexity of Nonrecursive Logic Programs with Complex Values
MPI-I-97-2-009	A. Bockmayr, F. Eisenbrand	On the Chvátal Rank of Polytopes in the 0/1 Cube
MPI-I-97-2-008	A. Bockmayr, T. Kasper	A Unifying Framework for Integer and Finite Domain Constraint Programming
MPI-I-97-2-007	P. Blackburn, M. Tzakova	Two Hybrid Logics
MPI-I-97-2-006	S. Vorobyov	Third-order matching in $\lambda\rightarrow$-Curry is undecidable
MPI-I-97-2-005	L. Bachmair, H. Ganzinger	A Theory of Resolution
MPI-I-97-2-004	W. Charatonik, A. Podelski	Solving set constraints for greatest models
MPI-I-97-2-003	U. Hustadt, R.A. Schmidt	On evaluating decision procedures for modal logic
MPI-I-97-2-002	R.A. Schmidt	Resolution is a decision procedure for many propositional modal logics