Automatic Layout and Labelling of State Diagrams

Gunnar W. Klau[1] and Petra Mutzel[2]

[1] Max Planck Institute for Computer Science, Algorithms and Complexity Group, Stuhlsatzenhausweg 85, D-66123 Saarbrücken, Germany; currently Vienna University of Technology. E-Mail: guwek@mpi-sb.mpg.de
[2] Vienna University of Technology, Algorithms and Data Structures Group, Favoritenstr. 9-11, A-1040 Wien, Austria. E-Mail: mutzel@apm.tuwien.ac.at

Abstract. We consider the problem of automatically generating readable layouts for state diagrams. Such diagrams appear in the field of automation engineering in the design process of control systems. Our industrial partner, the Siemens AG, realised that due to the complex nature of these diagrams, automatic layout tools lead to a better design and documentation of control systems.

The layout problem turns out to be difficult, since not only a graph drawing problem has to be solved but also an additional labelling problem. In this article we study the combined graph layout and labelling problem and present new results for the two-dimensional compaction problem in graph drawing, the label number maximisation problem and the combined graph labelling problem.

1 Introduction

State diagrams are used for designing and running control systems like, *e.g.*, production controls or robot controls in the area of automation engineering. Figure 1 illustrates a typical hand-drawn state diagram of a control system produced at our industrial partner, the Siemens AG.

A state diagram describes all possible states of a control system and the state transitions. A state can be, *e.g.*, the initial state, an error state or states like “motor is running” and is displayed by a circle which is also called a *state node* in the diagram. A transition from a state A to state B is displayed as a connection line from state node A to state node B via a *transition node*. Rectilinear labels which are attached to the state and transition nodes contain further information—in many cases several lines of program code.

The drawing of Fig. 1 is very hard to read and understand, as many lines cross each other, labels often overlap with transition lines and many labels are placed far away from their associated nodes. Because of the complex nature of state diagrams, our industrial partner asked us to provide a tool for automatically generating readable layouts of state diagrams. First, we developed a prototype that used state-of-the-art layout techniques for graph drawing[1].

[1] Note that a state diagram can be seen as a graph in which the nodes correspond to the states and transitions, and the edges correspond to the transition lines.

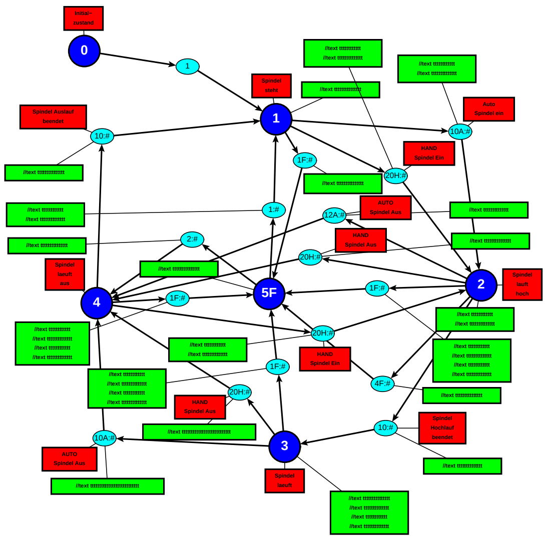


Fig. 1. A hand-drawn state diagram

Figure 2 shows the same state diagram as Fig. 1, now semi-automatically produced by our prototype. We first used a graph layout algorithm and then a method for map labelling. This approach forces some labels to either overlap parts of the drawing or places them far away from their associated nodes.

We realised that this approach—first draw, then label—will not lead to nice layouts for state diagrams. When drawing the graph, the appropriate space for the labels should already be reserved. This leads to combining the graph drawing and the map labelling problem. So far, in the literature, these problems have been treated separately.

The research aim of this project was to investigate the combined drawing and labelling problem in order to produce readable automatically generated layouts of state diagrams. For drawing the graphs we use the so-called *topology-shape-metrics paradigm* (see, *e.g.*, [5]). This approach produces nice drawings with only a small number of crossings in which the line segments are mainly orthogonal, *i.e.*, horizontal or vertical. Here, a first step (crossing minimisation problem) determines the topology of the layout so that the number of crossings is small. A second step (bend minimisation problem) fixes the shape of the layout, *i.e.*, essentially the bends and the angles, trying to introduce as few bends in the edges as possible. In a third step (compaction problem), the lengths of the line segments are calculated. Here, the objective is to minimise the total edge length or the area.

In this scenario, it seems natural to investigate the combination of the compaction problem with the labelling problem. We call this problem a *graph labelling* problem in contrast to a problem from map labelling where the co-

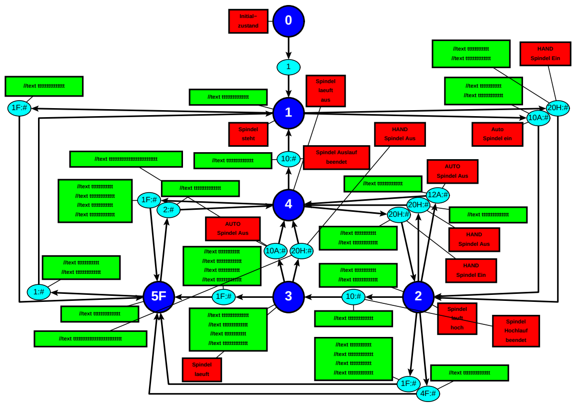


Fig. 2. The example from Fig. 1. First compute an orthogonal layout, then solve a map labelling problem

ordinates of the points are fixed in the plane. In Sect. 2 we give the mathematical formulations of all problems we consider in this article.

We first investigated the compaction problem. Formally, the pure two-dimensional compaction problem for orthogonal graph drawing is to determine coordinates for the vertices and bends, so that the total edge length is minimised and the shape of the drawing is preserved. This problem is closely related to problems in VLSI-design and has been shown to be *NP*-hard. We have found a new graph-theoretical formulation for the compaction problem. In [13], we have characterised 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 which code horizontal and vertical positioning relations. 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 has been 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 minimised. We can recognise those instances having only one such extension; for these cases we solve the compaction problem in polynomial time.

Hence, we have transformed the compaction problem, which is of rather geometrical nature, into a graph-theoretical one which can naturally be formulated as an integer linear program. Our computational experiments show that the approach works well in practice. Section 3 deals with the pure compaction problem.

Our idea of using constraint graphs for the compaction problem could be transformed to the map labelling problem. Again, a pair of constraint graphs in x- and y-direction is linked by a set of additional constraints, thus characterising all feasible solutions of the label number maximisation problem. Formally, the label number maximisation problem asks for the maximum number of labels from a given set that can be placed in the plane so that each such label is attached to its corresponding point and no two labels overlap. Using our new approach, we can not only express various discrete models (*e.g.*, the four-position model in which one of the label corners is attached to the point) but also the slider labelling model. The slider model allows a continuous movement of a label around its point feature, leading to a significantly higher number of labels that can be placed. To our knowledge, we have presented the first algorithm that computes provably optimal solutions in the slider model. Our experimental results on instances created by a widely used benchmark generator indicated that the new approach is applicable in practice. See Sect. 4 for a detailed description.

Section 5 shows that it is possible to integrate both approaches into one in order to attack the combined compaction and labelling problem. Formally, the graph labelling problem can be stated as follows: Given an orthogonal representation—as produced by algorithms within the topology-shape-metrics paradigm—the task is to generate a labelled orthogonal embedding with minimum total edge length.

Unfortunately, so far we have not been able to transfer our theoretical results for the combined problem into practice. Our first experimental results using a simple prototype are promising, however. For the remaining project time, we will work on integrating our approach into a practical tool, which can be used by our industrial partner to generate nice and readable layouts for state diagrams. In Sect. 6 we summarise our results for the combined compaction and labelling problem.

2 Mathematical Problem Formulation

As mentioned in the introduction, we concentrate on the compaction phase of the topology-shape-metrics paradigm in order to assign lengths and place labels at the same time. In this section, we present precise mathematical formulations—first for the two subproblems, and then for the combined compaction and labelling problem. We assume familiarity with planarity and basic graph theory.

The output of the second phase within the paradigm, orthogonalisation, is an *orthogonal representation* H which contains the information about the planar topology and the shape of the drawing. We call an orthogonal drawing *simple* if its number of bends is zero[2]. 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. Note that the notion of shape induces a partitioning of drawings in equivalence classes. Two orthogonal drawings belong to the

[2] We can always assume that an orthogonal representation is simple by treating bends as artificial vertices.

same class if one can be obtained from the other by modifying the lengths of the horizontal and vertical edge segments without changing the angles formed by them. Formally, for a simple orthogonal drawing, the orthogonal representation 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.

In order to compute a drawing for H, we must assign coordinates to the vertices. In this article, we concentrate on *pure orthogonal embeddings.* They are only admissible for 4-planar graphs, *i.e.*, planar graphs whose maximum vertex degree does not exceed four. A pure orthogonal embedding maps vertices and bends to distinct points and edge segments to horizontal or vertical non-crossing line segments of some minimum length connecting the images of their endpoints. As with representations, we call orthogonal embeddings simple if they do not contain bends. A special case are *pure orthogonal grid embeddings* as defined in [17]: Here, vertices and bends must have integer coordinates and the minimum length equals one. Our ideas can however be adapted to other drawing standards such as *Kandinsky-like embeddings* (introduced in [7]) or *orthogonal box embeddings*, used, *e.g.*, by the Giotto algorithm ([18]); subclasses of this standard are the *big node model* from [8] and the *TSS* model from [1], a related class are the *quasi-orthogonal embeddings* as defined in [11]. We state the pure two-dimensional compaction problem as follows:

Problem 1 (Compaction problem for orthogonal drawings, COMP*).* Given an orthogonal representation H, find a pure orthogonal grid embedding Γ for H of minimum total edge length.

Patrignani has shown *NP*-hardness of COMP in [16]. The second subproblem is the labelling task. Let R be the set of axis-parallel rectangles in the plane. A *labelling* for a set $P = \{p_1, \dots, p_k\}$ of k points in the plane, a set $L = \{\lambda_1, \dots, \lambda_l\}$ of l labels, two functions $w, h : L \to \mathbb{Q}$ and a function $a : L \to P$ is an assignment $r : L \to R$, so that the following conditions hold:

(L1) Rectangle $r(\lambda)$ has width $w(\lambda)$ and height $h(\lambda)$ for every $\lambda \in L$.
(L2) Point $a(\lambda)$ lies on the boundary of $r(\lambda)$ for all $\lambda \in L$.
(L3) The open intersection $r(\lambda) \cap r(\mu)$ is empty for all distinct $\lambda, \mu \in L$.

Properties (L1) and (L2) make sure that each label is attached correctly to its point feature and drawn with the given size. Property (L3) prohibits overlaps between the labels. We allow, however, that two labels touch each other. An interesting version of the labelling subproblem is its maximisation variant:

Problem 2 (Label Number Maximisation, LNM*).* Given a set $P = \{p_1, \dots, p_k\}$ of k points in the plane, a set $L = \{\lambda_1, \dots, \lambda_l\}$ of l labels, two functions $w, h : L \to \mathbb{Q}$ and a function $a : L \to P$, find a labelling for a subset $L_P \subseteq L$ of largest cardinality.

Also LNM is NP-hard in most axis-parallel labelling models[3]. We can now formally state the combined compaction and labelling problem. A *labelled orthogonal embedding* Γ_L for an orthogonal representation H of a planar graph $G = (V, E)$, a label set L, size functions $w, h : L \to \mathbb{Q}$ and a function $a : L \to V$ is a tuple (Γ, r) where Γ is an orthogonal embedding of H and r is a labelling for L with point set $\{\Gamma(v) \mid v \in V\}$.

Problem 3 (Compaction and Labelling, COLA*).* Given an orthogonal representation H for a planar graph $G = (V, E)$, a label set L with functions $w, h : L \to \mathbb{Q}$, and $a : L \to V$, find a labelled orthogonal embedding for H with minimum total edge length.

Note that in this article we have included proofs only if they are especially important or necessary for understanding our techniques, *e.g.*, constructive proofs building the basis of algorithms.

3 Optimal Two-Dimensional Compaction

In this section we present our results for the pure two-dimensional compaction problem COMP. We also introduce the combinatorial framework which will be needed throughout the rest of this article. A detailed discussion of our approach to solving the compaction problem to optimality appeared in [13], a full version of the paper is [10].

So far, in graph drawing only heuristics have been used for compacting orthogonal grid drawings. Tamassia suggested in [17] refining the shape of the drawing into one with rectangular faces by introducing artificial vertices and edges. If all the interior faces are rectangles, COMP 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 vertices and edges. Another method with better results but the same drawback is given in [2]. Of course, once a first sketch of the graph has been computed, VLSI-based techniques can be used to further improve the quality of the layout. In one-dimensional compaction, the goal is to minimise the width or height of the layout while preserving the coordinates of the fixed dimension. The one-dimensional version of the compaction problem can be solved in polynomial time using the so-called *layout graphs.* Figure 3 illustrates the differences in the output of four different compaction methods for a given graph with fixed shape.

But the result of this further improvement is also often not satisfying. Figure 4(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 these methods are based on visibility properties. A

[3] The models are defined by further restrictions of property (L2), see Sect. 4 for details. For the unrestricted version as above, a complexity proof appears in [19].

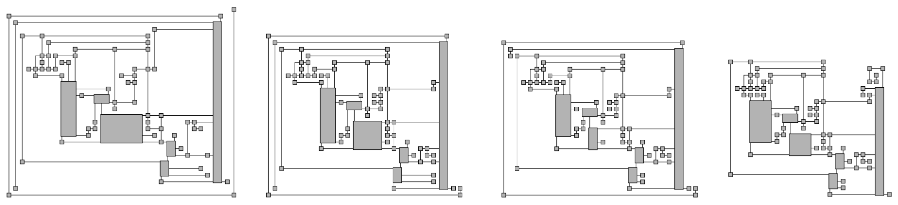


Fig. 3. The result of four different compaction algorithms: traditional, longest paths, flow, optimal

two-dimensionally optimal solution is Fig. 4(b) in which the total edge length is only $k + 6$.

3.1 The Combinatorial Framework

Initial observation leads to the following notion: In an orthogonal drawing, incident edges of same direction can be combined; the emerging sets of consecutive edges form the objects to compact. Let Γ be a simple orthogonal 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 (respectively vertical) *subsegment* in Γ is a connected component in (V, E_h) (respectively (V, E_v)). If the component is maximally connected it is also referred to as a *segment*. Three observations are crucial (also see Fig. 5).

1. Each edge is a subsegment.
2. Each vertex v belongs to one unique horizontal and one unique vertical segment, denoted by $\mathrm{hor}(v)$ and $\mathrm{vert}(v)$.
3. The *limits* of a subsegment s are given as follows: Let v_l, v_r, v_b, and v_t be the leftmost, rightmost, bottommost, and topmost vertices on s. Then $l(s) = \mathrm{vert}(v_l)$, $r(s) = \mathrm{vert}(v_r)$, $b(s) = \mathrm{hor}(v_b)$, and $t(s) = \mathrm{hor}(v_t)$.

Let S_h and S_v denote the set of horizontal and vertical segments, respectively. The following lemma implies that, in an orthogonal drawing for a graph $G = (V, E)$, the total number of segments is $2|V| - |E|$.

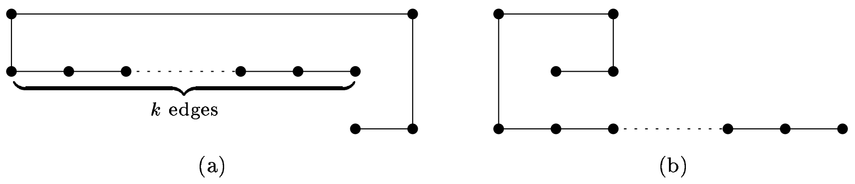
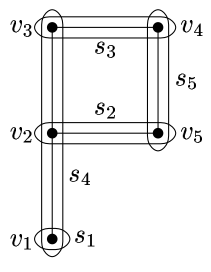


Fig. 4. A drawing generated with (a) an iterative one-dimensional and (b) an optimal compaction method



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

i	$V(s_i)$	$E(s_i)$	$l(s_i)$	$r(s_i)$	$b(s_i)$	$t(s_i)$
1	$\{v_1\}$	$\{\}$	s_4	s_4	s_1	s_1
2	$\{v_2, v_5\}$	$\{(v_2, v_5)\}$	s_4	s_5	s_2	s_2
3	$\{v_3, v_4\}$	$\{(v_3, v_4)\}$	s_4	s_5	s_3	s_3
4	$\{v_1, v_2, v_3\}$	$\{(v_1, v_2), (v_2, v_3)\}$	s_4	s_4	s_1	s_3
5	$\{v_4, v_5\}$	$\{(v_4, v_5)\}$	s_5	s_5	s_2	s_3

Fig. 5. Segments of a simple orthogonal grid drawing and its limits

Lemma 4. *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 $|S_v| = |V| - |E_v|$.*

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.* Nodes in these graphs represent the segments; their weighted arcs characterise relative positioning relations between the segments. We denote by $c(s_i)$ the coordinate of segment s_i. A weight $\omega_{ij} \in \mathbb{Q}$ for an arc (s_i, s_j) indicates that the coordinate difference $c(s_j) - c(s_i)$ must be at least ω_{ij}. Unlike the layout graphs which are based on visibility properties, the pair of constraint graphs arises from the shape of the drawing.

Definition 5. A *coordinate assignment* for a pair of constraint graphs (D_x, D_y) with $D_x = (V_x, A_x)$, $D_y = (V_y, A_y)$ and arc weights $\omega \in \mathbb{Q}^{|A_x \cup A_y|}$ is a function $c : V_x \cup V_y \to \mathbb{Q}$. We say c *respects* an arc set $A \subseteq A_x \cup A_y$ if $c(v_j) - c(v_i) \geq \omega_{ij}$ for all $(v_i, v_j) \in A$.

We will show later that we can use a pair of constraint graphs together with a coordinate assignment which respects this pair to construct a feasible orthogonal grid embedding—as long as the graphs satisfy certain conditions. The following theorem expresses an important connection between constraint graphs and coordinate assignments.

Theorem 6. *Let $D = (V, A)$ be a constraint graph with arc weights $\omega \in \mathbb{Q}^{|A|}$. A coordinate assignment c that respects A exists if and only if A does not contain a directed cycle of positive weight.*

Proof. For the forward direction assume otherwise. Without loss of generality, let $C = \big((v_1, v_2), (v_2, v_3), \ldots, (v_{|C|}, v_1)\big)$ be a directed cycle of positive weight in A. Since c respects A, it must respect in particular the arcs in C. It follows that $c(v_2) - c(v_1) \geq \omega_{(1,2)}$, $c(v_3) - c(v_2) \geq \omega_{(2,3)}, \ldots$, $c(v_1) - c(v_{|C|}) \geq \omega_{(|C|,1)}$. Summing up the left sides yields zero, summing up the right sides yields $\sum_{a \in C} \omega_a$. We get $0 \geq \sum_{a \in C} \omega_a > 0$, a contradiction.

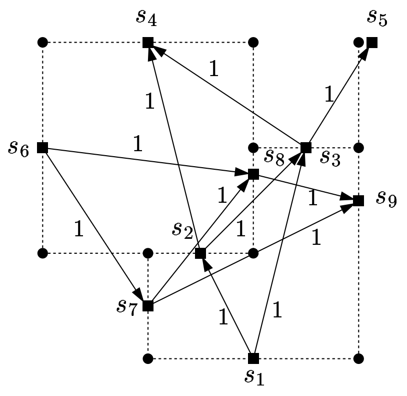


Fig. 6. An orthogonal grid embedding (dotted) and a pair of constraint graphs

For the backward direction of the proof, let $\mathcal{A}\, c \geq \omega$ be the set of inequalities describing the requirements for a coordinate assignment. By Farkas' Lemma, there is a feasible solution if and only if no vector $y \geq 0$ with $y^T \mathcal{A} = 0$ and $y^T \omega > 0$ exists. Assume otherwise, *i.e.*, there is such a vector y. Then y corresponds to a flow in A with positive weight. Since all supplies in the corresponding network are zero, this flow must be circular and thus corresponds to a directed cycle of positive weight. □

Figure 6 shows an example of a pair of constraint graphs. The arcs specify exactly the relative relationships known from the shape of the graph: We call such special pairs of constraint graphs $((S_v, A_h),\ (S_h, A_v))$ *shape graphs* or a *shape description.* Note that each horizontal edge in the original graph defines a relative positioning constraint between two vertical segments. Similarly, vertical edges determine constraints between horizontal segments.

The characterisation of feasible solutions for the two-dimensional compaction problem is based on the following three observations:

1. The arcs of the shape graphs are contained in every pair of constraint graphs corresponding to a drawing reflecting the given shape.
2. Generally, the information in the shape graphs is not sufficient to produce an orthogonal embedding. If this is the case, however, we will see that they fulfil a special property which we will refer to as *completeness.*
3. There are in general many possibilities for extending the shape graphs to a complete pair of constraint graphs.

We denote by $s_i \xrightarrow{+} s_j$ a path of positive weight between s_i and s_j. The following is a precise characterisation for complete pairs of constraint graphs in terms of paths that must be contained in the arc sets.

Definition 7. A pair of constraint graphs is *complete* if and only if both arc sets do not contain positive cycles and for every pair of segments $(s_i, s_j) \in S \times S$ one of the following four conditions holds:

1. $r(s_i) \xrightarrow{+} l(s_j)$
2. $r(s_j) \xrightarrow{+} l(s_i)$
3. $t(s_j) \xrightarrow{+} b(s_i)$
4. $t(s_i) \xrightarrow{+} b(s_j)$

If one of the positive paths in Def. 7 exists we call the pair of segments (s_i, s_j) *separated*[4]. We can now express a one-to-one correspondence between these complete extensions and orthogonal embeddings. On the basis of the following theorem, the two-dimensional compaction task can be seen as the search for a complete extension of the given shape graphs leading to minimum total edge length.

Theorem 8. *For each simple orthogonal embedding with shape description* $\sigma = ((S_v,\ A_h),\ (S_h, A_v))$ *there exists a complete extension* $\tau = ((S_v, B_h),$ $(S_h, B_v))$ *of* σ *and vice versa.*

Proof (Sketch). To prove the first part of the theorem, we consider a simple orthogonal grid drawing Γ with shape description $\sigma = ((S_v, A_h), (S_h, A_v))$. Let $c(s_i)$ denote the fixed coordinate for segment $s_i \in S_h \cup S_v$. We construct a complete extension $\tau = ((S_v, B_h), (S_h, B_v))$ 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 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 prohibits 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. Any topological sorting algorithm, *e.g.*, one based on longest paths computations in the acyclic graphs in τ, computes an integer coordinate assignment c as defined in Def. 5 for τ. Given c, 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. □

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 minimises the total edge length. If a shape description already

[4] In the special case that all arc weights equal one—as in the problem COMP—the notion of completeness is simpler: Both arc sets must be acyclic and for every segment pair one of the four paths must be present.

satisfies the conditions of a complete extension (see Fig. 7(a)), the compaction problem can be solved optimally in polynomial time: The resulting problem is the dual of a minimum cost flow problem. Sometimes the shape description is not complete but it is only possible to extend it in one way (see Fig. 7(b)). In these cases it is also 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. 7(c)).

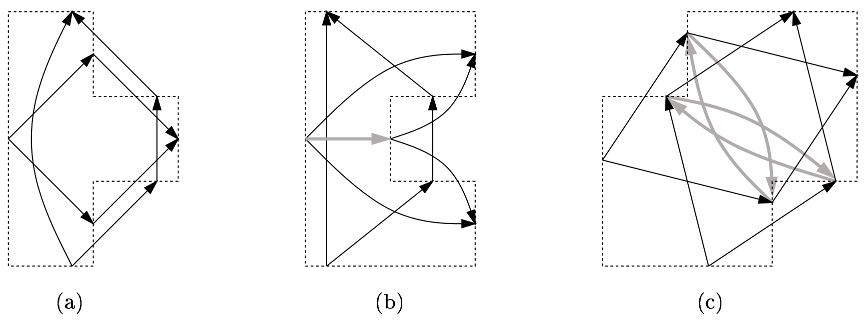


Fig. 7. Three types of shape descriptions. Dotted lines show the orthogonal grid drawings, thin arrows arcs in shape descriptions and thick gray arcs possible completions

The graph-based characterisation can be used to obtain an integer linear programming formulation for the problem COMP. Since the pure compaction problem is the special case of the combined compaction and labelling problem in which the label set is empty, we defer the presentation of our approach until Sect. 5 and mention at this point some of our experimental results for the pure compaction problem.

3.2 Computational Results

We compare the results of our method—which we will refer to as OPT—to the results achieved by two other compaction methods: ORIG is an implementation of the traditional method proposed in [17]. 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 phase. 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, first mentioned in [6], contains more than 11,000 graphs representing data from real-world applications. For our experimental study, we planarise each of the graphs by computing a planar subgraph and reinserting the edges, representing each crossing by a virtual vertex. After fixing the planar embedding for every graph we compute its shape using an extension of Tamassia's bend minimising algorithm, presented in [11]. The resulting orthogonal representation is the input for the three compaction algorithms. We compare the total edge length and the area of the smallest enclosing rectangle of the drawings produced by ORIG, 1DIM, and OPT. Furthermore, we record their running times.

All the examples can be solved to optimality on a SUN Enterprise 10000. For all instances, the running times of ORIG and 1DIM are below 0.05 and 0.43 seconds, respectively. OPT solves the vast majority of instances (95.5%) in less than one second, few graphs need more than five seconds (1.1%) and only 29 graphs need more than one minute. The longest running time, however, is 68 minutes.

The average improvement of the total edge length computed by OPT over the whole test set of graphs is 2.4% compared to 1DIM and 21.0% compared to ORIG. Just looking at hard instances where OPT needs more than two seconds of running time we yield average improvements of 7.5% and 36.1%, respectively. Figures 8 and 9 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 compute the minimal, maximal and average improvement for the graphs of the same size. 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 in some cases OPT yields improvements of more than 30% in comparison to the previously best strategy; Fig. 3 at the beginning of this section shows the drawings for such a case (here, the improvement is 28%). For the area, the data look similar with the restriction that in a few cases the values produced by 1DIM are slightly better than those from OPT: short edge length does not necessarily lead to small area. The average area improvements compared to 1DIM and ORIG are 2.7% and 29.3%, however.

In general, we make the following observations: Instances of COMP divide into easy and hard problems, depending on 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 are hard to compact since their number of fundamentally different drawings is in general very high, but in these cases the improvement is much greater.

4 Optimal Labelling in the Slider Model

In this section we investigate the label number maximisation problem LNM as defined in Sect. 2. Like the pure compaction problem, this problem is

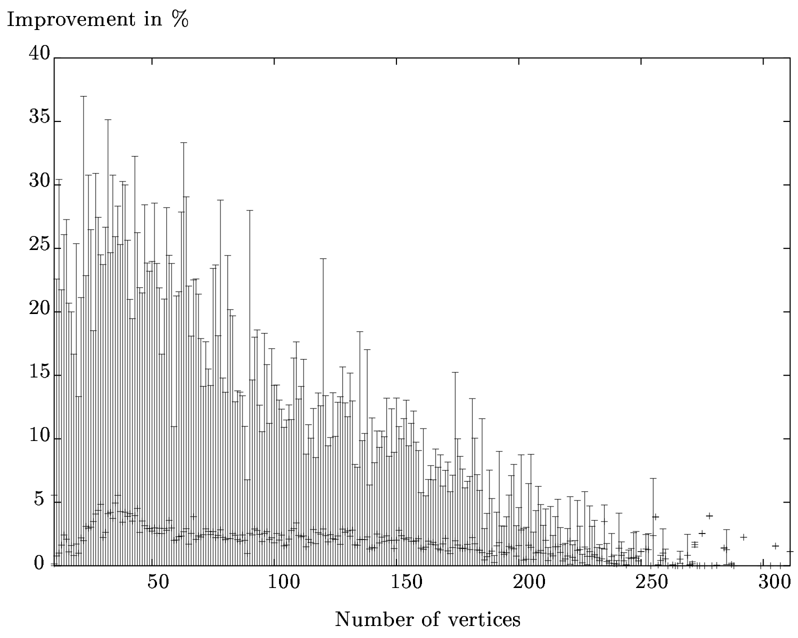


Fig. 8. Quality of OPT compared to 1DIM

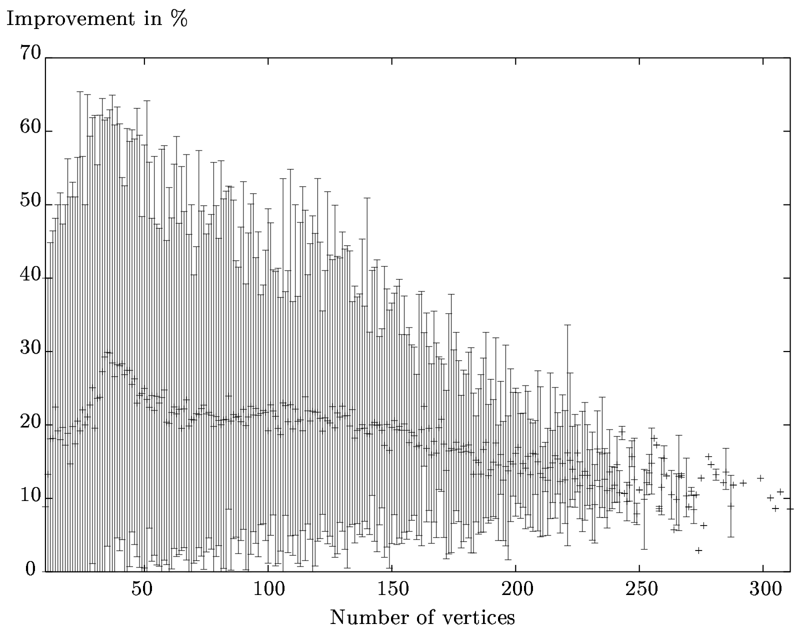


Fig. 9. Quality of OPT compared to ORIG

interesting in its own right and has direct applications, *e.g.*, in cartography, geographical information systems and graphical interfaces. A detailed description of our approach is [14].

Many papers have been published on the label number maximisation problem (for an overview, see the bibliography [21]). So far, most previous work on map labelling has concentrated on the discrete model, which allows only a finite number of positions per label. The most popular discrete model is the four-position model (see Fig. 10); the two- and one-position models have been introduced rather for theoretical purposes. In [4], Christensen, Marks and Shieber have presented a comprehensive treatment of LNM in the four-position model including complexity analysis, heuristic methods and a computational study. They have introduced a procedure for randomly creating labelling instances which has become a widely used benchmark generator in the map labelling literature. The only practically efficient algorithm for computing provably optimal solutions in the discrete model has been suggested by Verweij and Aardal in [20]. They treat the problem as an independent set problem and solve it using a branch-and-cut algorithm. The algorithm is able to optimally label up to 800 point features (using the benchmark generator from [4]) within moderate computation time (about 20 minutes).

More natural than the discrete model is the slider model, which allows a continuous movement of a label around its point feature. Although Hirsch already considered this model in 1982, it was not further investigated until very recently. In [19], van Kreveld, Strijk and Wolff introduce several variations of the slider model (see Fig. 10). They prove *NP*-hardness of LNM in the four-slider model and suggest a polynomial time algorithm which is able to find a solution that is at least half as good as an optimal solution. Moreover, their computational results show that the slider model is significantly better than the discrete model. The four-slider model allows the placement of up to 15% more labels in real-world instances and up to 92% more labels in pseudo-random instances.

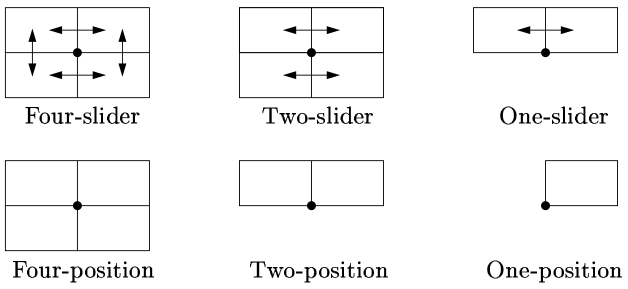


Fig. 10. Axis-parallel rectangular labelling models. A label can be placed in any of the positions indicated by the rectangles and slid in the directions of the arcs

In this section, we will present an algorithm for solving instances of LNM to optimality that works in any of the above mentioned labelling models. We allow several labels per point feature and labels of different sizes. Figure 11 shows a provably optimal labelling for 700 point features computed with a first implementation of our new approach; 699 labels could be placed. Like the algorithm for the compaction problem, it is based on the pair of constraint graphs defined in Sect. 3. In the following, we show how to treat labels inside our combinatorial framework.

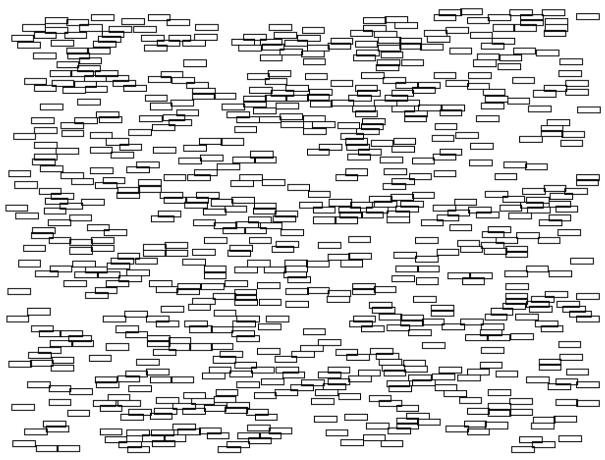


Fig. 11. Optimal labelling of 700 point features in the four-slider model. Instance randomly generated with a standardised procedure described in [4]

4.1 Extending the Combinatorial Framework

Modelling point features. The positions of the k point features are specified in the input set P. For each $p_i \in P$ with coordinates $x(p_i)$ and $y(p_i)$ we introduce a node x_i in V_x and a node y_i in V_y; one for its x-, one for its y-coordinate. We fix the positions of the point features by inserting four directed paths $P_x = (x_1, \ldots, x_k)$, $P_{-x} = (x_k, \ldots, x_1)$, $P_y = (y_1, \ldots, y_k)$ and $P_{-y} = (y_k, \ldots, y_1)$ with weights $\omega_{x_i x_{i+1}} = x(p_{i+1}) - x(p_i)$, $\omega_{x_{i+1} x_i} = x(p_i) - x(p_{i+1})$, $\omega_{y_i y_{i+1}} = y(p_{i+1}) - y(p_i)$ and $\omega_{y_{i+1} y_i} = y(p_i) - y(p_{i+1})$ for $i \in \{1, \ldots, k-1\}$.

We call the directed edges on these paths *fixed distance arcs* and refer to them as A_F. Figure 12 shows a set of point features and its representation in the constraint graphs.

Lemma 9. *A coordinate assignment c that respects A_F results in a correct placement of point features (up to translation).*

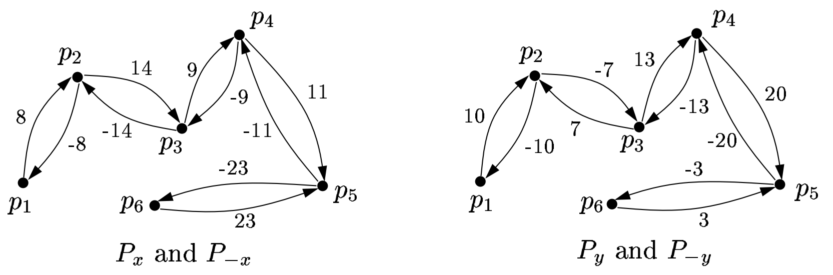


Fig. 12. Modelling the placement of point features with fixed distance arcs

Modelling labels. Each label $\lambda \in L$ has to be represented by a rectangle $r(\lambda)$ of width $w(\lambda)$ and height $h(\lambda)$. Additionally, we have to ensure that λ will be placed correctly with respect to its point feature $a(\lambda)$, *i.e.*, $a(\lambda)$ must lie on the boundary of $r(\lambda)$.

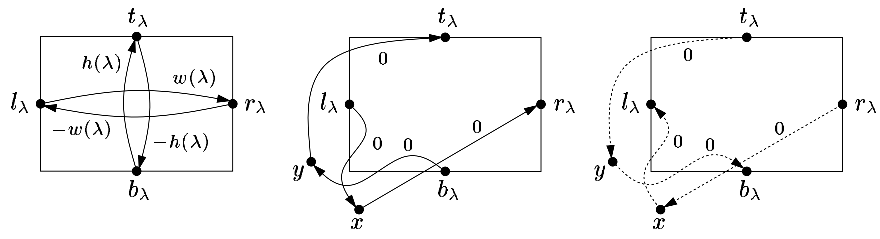
Straightforwardly, we model a label λ by two nodes in V_x and two nodes in V_y, representing the coordinates of $r(\lambda)$. Following the terminology in Sect. 3, we call these nodes the left, right, bottom and top *limit* of λ and refer to them as l_λ, r_λ, b_λ and t_λ, respectively. We introduce four *label size arcs* $A_S(\lambda) = \{(l_\lambda, r_\lambda), (r_\lambda, l_\lambda), (b_\lambda, t_\lambda), (t_\lambda, b_\lambda)\}$ in order to model the size of $r(\lambda)$. The weights of these arcs are $\omega_{l_\lambda r_\lambda} = w(\lambda)$, $\omega_{r_\lambda l_\lambda} = -w(\lambda)$, $\omega_{b_\lambda t_\lambda} = h(\lambda)$ and $\omega_{t_\lambda b_\lambda} = -h(\lambda)$, see Fig. 13(a).

A label λ must be placed close to its point feature $a(\lambda)$. Let x and y be the nodes representing point $a(\lambda)$ in the constraint graphs. We add four *proximity arcs* $A_P(\lambda) = \{(x, r_\lambda), (l_\lambda, x), (y, t_\lambda), (b_\lambda, y)\}$, as illustrated in Fig. 13(b). These arcs have zero weight and exclude the point feature $a(\lambda)$ from lying outside the rectangle $r(\lambda)$.

The point feature may still lie inside $r(\lambda)$. We disallow this by adding at least one of the four *boundary arcs* $\{(r_\lambda, x), (x, l_\lambda), (t_\lambda, y), (y, b_\lambda)\}$, each of weight zero. Note that these arcs are inverse to the proximity arcs for label λ. If, *e.g.*, (r_λ, x) is present in D_x, it forces—together with its inverse proximity arc (x, r_λ)—the coordinate of the right side of $r(\lambda)$ to be equal to the coordinate of x; the label has to be placed at its leftmost position. See also Fig. 13(c).

At this point we can influence the labelling model. We define a set $A_B(\lambda)$ containing the boundary arcs for the different models:

Labelling model		Boundary arcs $A_B(\lambda)$
Slider model	Four-slider	$\{(r_\lambda, x), (x, l_\lambda), (t_\lambda, y), (y, b_\lambda)\}$
	Two-slider	$\{(t_\lambda, y), (y, b_\lambda)\}$
	One-slider	$\{(y, b_\lambda)\}$
Discrete model	Four-position	$\{(r_\lambda, x), (x, l_\lambda), (t_\lambda, y), (y, b_\lambda)\}$
	Two-position	$\{(y, b_\lambda), (r_\lambda, x), (x, l_\lambda)\}$
	One-position	$\{(y, b_\lambda), (x, l_\lambda)\}$



(a) Label size arcs (b) Proximity arcs (c) Boundary arcs

Fig. 13. Modelling labels

For a slider model, at least one arc $a \in A_B(\lambda)$ must be contained in the constraint graph, for a discrete model at least two. *E.g.*, for the four-slider model, the set $A_B(\lambda)$ consists of all four boundary arcs, one of which must be present in the appropriate constraint graph. Note that we can express all six axis-parallel rectangular labelling models we have introduced at the beginning of this section as additional requirements on the constraint graphs.

Lemma 10. *Let λ be a label, c be a coordinate assignment respecting $A_S(\lambda)$, $A_P(\lambda)$ and at least d boundary arcs from $A_B(\lambda)$. Then c results in a placement in which λ is represented by a rectangle $r(\lambda)$ of width $w(\lambda)$ and height $h(\lambda)$. The label is placed so that point feature $a(\lambda)$ lies on the boundary of $r(\lambda)$ if $d = 1$ and on a corner of $r(\lambda)$ if $d = 2$.*

Avoiding label overlaps. To this point we have ensured that each label is placed correctly with respect to its point feature. We must still guarantee that the intersection of rectangles is empty. A crucial observation is that it suffices to consider only the pairs of labels that can possibly interact. If there is any overlap, such a pair must be involved.

Consider two different labels λ and μ and their corresponding rectangles $r(\lambda)$ and $r(\mu)$. We call the pair *vertically separated* if $r(\lambda)$ is placed either above or below $r(\mu)$. Similarly, λ and μ are *horizontally separated* if one rectangle is placed left to the other. Two labels overlap if they are neither vertically nor horizontally separated, we can exclude this by introducing one of the following four *label separation arcs* $A_S(\lambda, \mu) = \{(t_\mu, b_\lambda), (t_\lambda, b_\mu), (r_\mu, l_\lambda), (r_\lambda, l_\mu)\}$. Label separation arcs have zero weight.

Let R_λ be the boundary of the region in which label λ can be placed. Note that R_λ is defined by lower left corner $(x(a(\lambda)) - w(\lambda), y(a(\lambda)) - h(\lambda))$ and upper right corner $(x(a(\lambda)) + w(\lambda), y(a(\lambda)) + h(\lambda))$. Likewise, we determine R_μ for label μ. If the intersection of R_λ and R_μ is empty, λ and μ can never overlap, and we do not have to add any label separation arcs for this pair. In this case we set $A_S(\lambda, \mu) = \{\}$.

Consider now the case in which the intersection of R_λ and R_μ is not empty, as depicted in Fig. 14. Depending on the position of the corresponding point features $a(\lambda)$ and $a(\mu)$, $A_S(\lambda, \mu)$ contains the following label separation arcs:

1. If $x(a(\mu)) \geq x(a(\lambda))$ we have $(r_\lambda, l_\mu) \in A_S(\lambda, \mu)$.
2. If $x(a(\lambda)) \geq x(a(\mu))$ we have $(r_\mu, l_\lambda) \in A_S(\lambda, \mu)$.
3. If $y(a(\mu)) \geq y(a(\lambda))$ we have $(t_\lambda, b_\mu) \in A_S(\lambda, \mu)$.
4. If $y(a(\lambda)) \geq y(a(\mu))$ we have $(t_\mu, b_\lambda) \in A_S(\lambda, \mu)$.

Note that the only case in which $A_S(\lambda, \mu)$ contains all four label separation arcs occurs if λ and μ label the same point feature, *i.e.*, $a(\lambda) = a(\mu)$.

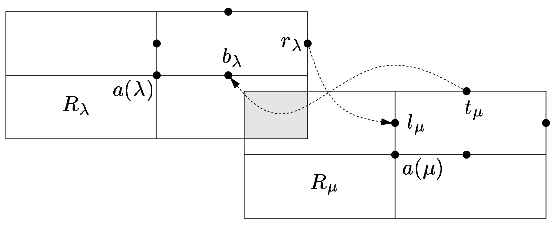


Fig. 14. Label separation arcs between two labels λ and μ

Lemma 11. *Let λ and μ be two labels that can possibly overlap and let c be a coordinate assignment respecting $A_S(\lambda)$, $A_S(\mu)$ and $A \subseteq A_S(\lambda, \mu)$ with $|A| \geq 1$. Then c results in a placement in which the two rectangles $r(\lambda)$ and $r(\mu)$ do not overlap.*

We refer to the boundary and label separation arcs as *potential arcs*

$$A_{\text{pot}} = \bigcup_{\lambda \in L} A_B(\lambda) \cup \bigcup_{\lambda, \mu \in L, \lambda \neq \mu} A_S(\lambda, \mu)$$

and state the label number maximisation problem in a combinatorial way. The task is to choose additional arcs from A_{pot} for a maximum number of labels without creating positive directed cycles.

Problem 12 (Constraint Graph Fulfilment problem, CGF*).* Given an instance of LNM, let $(D_x,\ D_y)$ be the pair of constraint graphs including only fixed distance arcs, label size arcs and proximity arcs. Let $d = 1$ if in the slider model and $d = 2$ if in the discrete model and let A_x and A_y be the arc sets of D_x and D_y, respectively. Find a set $L_P \subseteq L$ of greatest cardinality and an arc set $A \subseteq A_{\text{pot}}$ with the following properties:

(F1) $|A \cap A_B(\lambda)| \geq d$ for all $\lambda \in L_P$.
(F2) $|A \cap A_S(\lambda, \mu)| \geq 1$ for all $\lambda, \mu \in L_P$, $\lambda \neq \mu$, $R_\lambda \cap R_\mu \neq \emptyset$.
(F3) $A \cap A_x$ and $A \cap A_y$ do not contain a positive cycle.

Theorem 13. *Problems* LNM *and* CGF *are polynomially equivalent.*

Proof. Let A be a solution of CGF. We extend (D_x, D_y) by adding A to the arc sets. Because of (F3) and Thm. 6, there is a coordinate assignment that respects both the horizontal and vertical arc set. Lemmas 9, 10 and 11 ensure that properties (L1), (L2) and (L3) are fulfilled, thus we have a solution for LNM.

For the other direction, we start with the given coordinate assignment c resulting from the placement of labels. We create the set A of additional arcs as follows: For each label λ we add one or two boundary arcs, depending on how $r(\lambda)$ is placed with respect to point feature $a(\lambda)$. Similarly, we add appropriate arcs from $A(\lambda, \mu)$ for pairs of labels λ and μ, depending on the relative position of λ and μ in the labelling. Note that we have chosen the additional arcs so that they are respected by c. Properties (F1) and (F2) follow by construction, property (F3) follows by Thm. 6. □

We propose a zero-one integer linear programming formulation for solving CGF. The goal is to find the set of additional boundary and label separation arcs A and to determine which labels are to be placed.

We introduce two types of binary variables for this task: For each label λ there is a variable $y_\lambda \in \{0, 1\}$, indicating whether λ will be placed ($y_\lambda = 1$) or not ($y_\lambda = 0$). Additionally, there is a variable $x_a \in \{0, 1\}$ for each potential additional arc $a \in A_{\text{pot}}$ which is one if a is active and zero otherwise. Recall that d determines the labelling model—we have $d = 1$ in the slider and $d = 2$ in the discrete model.

We present the zero-one integer linear program (1) and show that it corresponds to CGF. We define $C_p = A_{\text{pot}} \cap C$.

$$
\begin{array}{llrr}
\max & \sum_{\lambda \in L} y_\lambda & & (1) \\
\text{subject to} & \sum_{a \in A_B(\lambda)} x_a - d y_\lambda \geq 0 & \forall \lambda \in L & (1.1) \\
& \sum_{a \in A_S(\lambda,\mu)} x_a - y_\lambda - y_\mu \geq -1 & \forall \lambda, \mu \in L, \lambda \neq \mu & (1.2) \\
& \sum_{a \in C_p} x_a \leq |C_p| - 1 & \forall \text{ positive cycles } C & (1.3) \\
& y_\lambda \in \{0, 1\} & \forall \lambda \in L & (1.4) \\
& x_a \in \{0, 1\} & \forall a \in A_{\text{pot}} & (1.5)
\end{array}
$$

We refer to (1.1) as *boundary inequalities*, to (1.2) as *label separation inequalities* and to (1.3) as *positive cycle inequalities.*

Theorem 14. *Each feasible solution (y, x) to (1) corresponds to a feasible solution of* CGF *and vice versa. The value of the objective function equals the cardinality of L_P.*

Corollary 15. *An optimal solution of (1) corresponds to an optimal labelling.*

Corollary 15 and Thm. 6 (the existence of a coordinate assignment) suggest an algorithm for attacking practical instances of the label number maximisation problem: In a first step, we solve (1) using integer programming techniques. The solution tells us which boundary and label separation arcs should be added to the arc sets of the constraint graphs (D_x, D_y). We use this information in a second step for computing the corresponding coordinate assignment via minimum cost flow (see proof of Thm. 6).

Our implementation is based on LEDA ([15]) and the ILP-solver in CPLEX ([9]). Due to the fact that there may be an exponential number of positive cycles in the constraint graphs, we use an iterative cutting plane approach. Our separation procedure uses the Bellman-Ford algorithm for detecting negative cycles applied to the constraint graphs after multiplying the arc weights $\omega \in \mathbb{Q}^{|A_x \cup A_y|}$ by -1. Our implementation is based on the one given in [3].

We have tested our algorithm on a widely used benchmark generator for randomly creating instances of LNM, following the rules described in [4]: First, we construct a set of n points with random coordinates in the range $\{0, \ldots, 792\}$ for the x- and $\{0, \ldots, 612\}$ for the y-coordinates. To each point feature belongs a label of width 30 and height 7. We have run the algorithm with both the slider and the discrete labelling models for rectilinear map labelling. Figure 11 in Sect. 1 shows an optimal solution for an instance with $n = 700$. In the optimal solution for the four-slider model, 699 labels can be placed, whereas at most 691 can be placed in the four-position model.

Evidently, more freedom in the labelling model results in a higher number of labels that can be placed. Two main factors influence the running time of our implementation for instances of the same size: On the one hand, this is the number of labels that cannot be placed in an optimal solution, *i.e.*, the difference $|L| - |L_P|$: To our surprise, we had the longest running times in the one-position model. On the other hand, the tightness of the inequalities seems to have an impact on the running time; the more restrictions on the variables, the faster the algorithms. Both factors, however, interrelate: In the more restricted models we can also place fewer labels.

5 Solving the Combined Problem

In this section, we combine the results from Sects. 3 and 4 and express the combined compaction and labelling problem COLA as a combinatorial optimisation problem in the pair of constraint graphs. More details can be found in

[12]. Furthermore, we present an integer linear program and show that each feasible solution corresponds to a solution for the COLA problem.

We have to extend the notion of shape description and completeness in order to integrate the labels: A *labelled shape description* σ_L of an orthogonal representation H with label information L, w, h, and a is a tuple $((S_v \cup V_{L_v}, A_h \cup A_{L_h}), (S_h \cup V_{L_h}, A_v \cup A_{L_v}))$ where $((S_v, A_h), (S_h, A_v))$ is a shape description for H and

$$V_{L_v} = \bigcup_{\lambda \in L} \{l_\lambda, r_\lambda\}, \qquad A_{L_h} = \bigcup_{\lambda \in L} (A_{S_h}(\lambda) \cup A_{P_h}(\lambda)),$$
$$V_{L_h} = \bigcup_{\lambda \in L} \{b_\lambda, t_\lambda\}, \qquad A_{L_v} = \bigcup_{\lambda \in L} (A_{S_v}(\lambda) \cup A_{P_v}(\lambda)) \ .$$

Here, $A_{S_h}(\lambda)$ and $A_{S_v}(\lambda)$ are the label size arcs and $A_{P_h}(\lambda)$ and $A_{P_v}(\lambda)$ correspond to the proximity arcs, both defined in Sect. 4, *i.e.*,

$$A_{S_h}(\lambda) = \{(l_\lambda, r_\lambda), (r_\lambda, l_\lambda)\} \qquad A_{S_v} = \{(b_\lambda, t_\lambda), (t_\lambda, b_\lambda)\}$$
$$A_{P_h}(\lambda) = \{(l_\lambda, r_{a(\lambda)}), (l_{a(\lambda)}, r_\lambda)\} \qquad A_{P_v} = \{(b_\lambda, t_{a(\lambda)}), (b_{a(\lambda)}, t_\lambda)\} \ .$$

As in the pure labelling problem, the label sizes determine the weights of arcs in A_{S_h} and A_{S_v}, the proximity arcs have zero weight. Observe that each instance of the COLA problem uniquely determines a labelled shape description. We can now give a more general formulation of completeness by considering not only the segments but also the labels. This generalisation leads to the subsequent main theorem whose proof is similar to the proof of Thm. 8.

Definition 16. A pair of labelled constraint graphs is *complete* if and only if the arc sets do not contain a directed cycle of positive weight and each distinct pair $o, p \in S_h \cup S_v \cup L, o \neq p$ is separated.

Theorem 17. *For each simple labelled orthogonal embedding with shape description* $\sigma = ((S_v, A_h), (S_h, A_v))$ *and label information* L, w, h, a *a complete labelled extension* $\tau = ((S_v \cup V_{L_v}, A_h \cup A_{L_h}), (S_h \cup V_{L_h}, A_v \cup A_{L_h}))$ *of* σ *exists and vice versa.*

5.1 Integer Linear Programming Formulation

We use this extended combinatorial framework to obtain an integer linear program formulation for the combined compaction and labelling problem COLA. Let an instance of the problem be characterised by the 4-planar graph $G = (V, E)$, a simple orthogonal representation H which divides E into E_h and E_v and the label information L, w, h and a. This determines the labelled shape description $\sigma_L = ((S_v \cup V_{L_v}, A_h \cup A_{L_h}), (S_h \cup V_{L_h}, A_v \cup A_{L_v}))$. Recall that the node set of this pair of constraint graphs is given by the set of segments $S = S_h \cup S_v$ and additional nodes $V_{L_v} \cup V_{L_h}$ which model the limits of the labels. The arc sets consist of the arcs in the shape description for H and the size and proximity arcs for the labels.

We consider the set of potential additional arcs A_{pot} that might be in some complete extension τ_L of σ_L. For a distinct pair $o, p \in S \cup L, o \neq p$, let $A_{\text{sep}}(o,p)$ be the arcs between the limits of o and p which ensure separation of the two objects (cf. Def. 7), so we have

$$A_{\text{pot}} = \bigcup_{o \neq p \in S \cup L} A_{\text{sep}}(o,p) \ .$$

As shown for the pure labelling problem and sketched for the pure compaction problem, a set $A_{\text{sep}}(o,p)$ rarely must have cardinality four, in most cases it can be empty. We introduce a variable $x_a \in \{0,1\}$ for each arc $a \in A_{\text{pot}}$ which is one if a is contained in the extension and zero otherwise. Additionally, we have a variable vector $c \in \mathbb{Q}^{|S \cup V_{L_h} \cup V_{L_v}|}$ to model the coordinate assignment for the pair of constraint graphs. Clearly, if we know c_v for all nodes v in the constraint graphs we can produce a labelled drawing of the graph with the given label information. Our integer linear programming formulation for the COLA problem is as follows (M is a very large number):

$$\min \sum_{e \in E_h} \left(c_{r(e)} - c_{l(e)}\right) + \sum_{e \in E_v} \left(c_{t(e)} - c_{b(e)}\right) \tag{2}$$

$$\text{subject to} \sum_{a \in A_{\text{sep}}(o,p)} x_a \geq 1 \qquad \forall o \neq p \in S \cup L \quad (2.1)$$

$$c_j - c_i \geq \omega_a \qquad \forall a = (i,j) \in A \quad (2.2)$$

$$c_j - c_i - (M + \omega_a)x_a \geq -M \qquad \forall a = (i,j) \in A_{\text{pot}} \quad (2.3)$$

$$x_a \in \{0,1\} \qquad \forall a = (i,j) \in A_{\text{pot}} \quad (2.4)$$

Inequalities (2.1) model the characterisation of separation, *i.e.*, the existence of necessary paths between distinct objects in an extension. Inequalities (2.2) force the coordinates to obey the distance rules coded by the weighted arcs in the underlying labelled shape description; the value ω_{ij} denotes the minimum distance between s_i and s_j. The same must hold true for the potential additional arcs: Whenever a variable x_a is one, *i.e.*, the potential arc a is part of the extension, we want an inequality of type (2.2), otherwise there should be no restriction on the coordinate variables. This situation is modelled by inequalities (2.3) with the help of a large constant M. We have shown in [13] that inequalities (2.2) and (2.3) ensure the absence of positive cycles in the extension.

Note that we do not introduce additional decision variables to integrate the labelling task; the choice of where to place a label is performed by the separation properties. A label can be placed wherever it does not interfere with other objects as long as it stays in the neighbourhood of the vertex to which it belongs. Moreover—as in the pure labelling problem—we do not restrict the label placement to a finite number of prescribed places. Of course, the number of possibly non-separated objects is much higher than in an instance of a pure compaction problem. Observe also that the boundary arcs, that serve in the pure labelling problem to force the boundary of a label to touch the appropriate point feature are a special case of the separation arcs.

Like the one-to-one correspondence between complete extensions and labelled orthogonal embeddings there is a one-to-one correspondence between feasible solutions of the integer linear program and complete extensions of the given labelled shape graphs:

Theorem 18. *For each feasible solution* (x, c) *of the integer linear program for a given labelled shape description* σ_L *there is a labelled orthogonal embedding with appropriate shape and label information and vice versa.*

Proof (Sketch). A solution of the integer linear program corresponds to a complete extension of the labelled shape description σ with appropriate length assignment. The forward direction follows with Thm. 8. Conversely, we can use the information of a labelled embedding to construct a feasible solution of the integer linear program. We can show that none of the inequalities is violated and that the value of the objective function equals the total edge length.

6 Conclusions

In order to devise an algorithm that automatically draws and labels state diagrams as they occur, *e.g.*, in automation engineering, we have studied the *NP*-hard combined compaction and labelling problem. This problem lies in the intersection of two research areas: *graph drawing* and *map labelling*. The task is to simultaneously assign consistent edge lengths for a given orthogonal representation which determines the shape of the drawing and to label subsets of vertices. The resulting drawing should have small total edge length and all labels should be placed at the corresponding vertices so that they do not overlap other objects. Already the two subproblems—the pure compaction task and the pure labelling task—are *NP*–hard.

We have introduced combinatorial formulations for the two-dimensional compaction problem and the maximisation variant of the labelling problem. Both subproblems have themselves interesting direct applications. We have given integer linear programming formulations for the combinatorial versions of the problems. Our experimental results show that this approach works well in practice—we can optimally compact graphs that have as many as 1,000 vertices and optimally label up to 700 point features in moderate computation time.

We have shown how to combine the combinatorial formulations of the two subproblems in order to model the combined compaction and labelling problem. We have presented an integer linear program to solve it, and preliminary computational experiments with instances from our industrial partner indicate that our approach can be successful in practice.

References

1. T. Biedl, B. Madden, and I. Tollis. The three-phase method: A unified approach to orthogonal graph drawing. In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *LNCS*, pages 391–402. Springer-Verlag, 1997.
2. S. Bridgeman, G. Di Battista, W. Didimo, G. Liotta, R. Tamassia, and L. Vismara. Turn-regularity and optimal area drawings for orthogonal representations. *Computational Geometry Theory and Applications (CGTA)*. To appear.
3. B. V. Cherkassky and A. V. Goldberg. Negative-cycle detection algorithms. *Mathematical Programming*, 85(2):277–311, 1999.
4. J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14(3):203–232, 1995.
5. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing.* Prentice Hall, 1999.
6. G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Computational Geometry: Theory and Applications*, 7:303–316, 1997.
7. U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. In F. J. Brandenburg, editor, *Graph Drawing (Proc. GD '95)*, volume 1027 of *LNCS*, pages 254–266. Springer–Verlag, 1996.
8. U. Fößmeier and M. Kaufmann. Algorithms and area bounds for nonplanar orthogonal drawings. In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *LNCS*, pages 134–145. Springer-Verlag, 1997.
9. ILOG. *CPLEX 6.5 Reference Manual*, 1999.
10. G. W. Klau and P. Mutzel. Optimal compaction of orthogonal grid drawings. Technical Report MPI-I-98-1-031, Max–Planck–Institut für Informatik, Saarbrücken, 1998.
11. 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, 1998.
12. G. W. Klau and P. Mutzel. Combining graph labeling and compaction. In J. Kratochvíl, editor, *Proc. 8th Internat. Symp. on Graph Drawing (GD '99)*, volume 1731 of *LNCS*, pages 27–37, Štiřín Castle, Czech Republic, 1999. Springer–Verlag.
13. G. W. Klau and P. Mutzel. Optimal compaction of orthogonal grid drawings. In G. P. Cornuéjols, R. E. Burkard, and G. J. Woeginger, editors, *Integer Programming and Combinatorial Optimization (IPCO '99)*, volume 1610 of *LNCS*, pages 304–319, Graz, Austria, 1999. Springer–Verlag.
14. G. W. Klau and P. Mutzel. Optimal labelling of point features in the slider model. In *Proc. 6th Annual International Computing and Combinatorics Conference (COCOON 2000)*, LNCS, Sydney, Australia, 2000. Springer–Verlag. To appear.
15. K. Mehlhorn and S. Näher. *LEDA. A Platform for Combinatorial and Geometric Computing.* Cambridge University Press, 1999.

16. M. Patrignani. On the complexity of orthogonal compaction. In F. Dehne, A. Gupta, J.-R. Sack, and R. Tamassia, editors, *Proc. 6th International Workshop on Algorithms and Data Structures (WADS '99)*, volume 1663 of *LNCS*, pages 56–61. Springer–Verlag, 1999.
17. R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987.
18. R. Tamassia, G. Di Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Trans. Syst. Man Cybern.*, SMC-18(1):61–79, 1988.
19. M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Computational Geometry: Theory and Applications*, 13:21–47, 1999.
20. B. Verweij and K. Aardal. An optimisation algorithm for maximum independent set with applications in map labelling. In *Proc. 7th Europ. Symp. on Algorithms (ESA '99)*, volume 1643 of *LNCS*, pages 426–437, Prague, Czech Republic, 1999. Springer–Verlag.
21. A. Wolff and T. Strijk. The map labeling bibliography. `http://www.math-inf.uni-greifswald.de/map-labeling/bibliography`.