Mathematical Programming manuscript No.
(will be inserted by the editor)

Gunnar W. Klau · Petra Mutzel

Optimal Labeling of Point Features in Rectangular Labeling Models*

the date of receipt and acceptance should be inserted later

Abstract. We investigate the *NP*-hard label number maximization problem (LNM): Given a set of rectangular labels Λ, each of which belongs to a point feature in the plane, the task is to find a *labeling* for a largest subset Λ_P of Λ. A labeling is a placement such that none of the labels overlap and each $\lambda \in \Lambda_P$ is placed according to a *labeling model*: In the *discrete models*, the label must be placed so that the labeled point coincides with an element of a predefined subset of corners of the rectangular label, in the *continuous* or *slider models*, the point must lie on a subset of boundaries of the label. Obviously, the slider models allow a continuous movement of a label around its point feature, leading to a significantly higher number of labels that can be placed.

We present exact algorithms for this problem that are based on a pair of so-called constraint graphs that code horizontal and vertical positioning relations. The key idea is to link the two graphs by a set of additional constraints, thus characterizing all feasible solutions of LNM. This enables us to formulate a zero-one integer linear program whose solution leads to an optimal labeling.

We can express LNM in both the discrete and the slider labeling models. To our knowledge, we present the first algorithm that computes provably optimal solutions in the slider models. We find it remarkable that our approach is independent of the labeling model and results in a discrete algorithm even if the problem is of continuous nature as in the slider models. Extensive experimental results on both real-world instances and point sets created by a widely used benchmark generator show that the new approach—although being an exponential time algorithm—is applicable in practice.

Key words. Map Labeling – Point Feature Map Labeling – Constraint Graphs – Combinatorial Optimization – Integer Programming

1. Introduction

Recently, map labeling has attracted a lot of researchers in computer science due to its numerous applications, *e.g.*, in cartography, geographic information systems and graphical interfaces. A major problem in map labeling is point-feature label placement. Here, the task is to place labels adjacent to point features so that no two labels overlap. In this paper, we focus on this problem and restrict the labels to be axis-parallel rectangles. For literature on other models, see the map labeling bibliography (Wolff and Strijk).

In general it is not possible to place all the given labels without any overlap. The literature suggests two possibilities to deal with this problem: decreasing the size of the labels so that all of them fit without any overlap, or keeping the sizes of the labels fixed while looking for the maximum number of labels that can be placed. The first

---

Gunnar W. Klau (e-mail: gunnar@ads.tuwien.ac.at) and Petra Mutzel (e-mail: mutzel@ads.tuwien.ac.at): Technische Universität Wien, Institute of Computer Graphics and Algorithms, Favoritenstr. 9-11, A-1040 Wien, Austria.

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

possibility is referred to as the *label size maximization problem*, the second one as the *label number maximization problem*.

In this paper we will concentrate on maximizing the number of labels. First, we define some notation about rectangles in the plane. Let $\mathbf{R}$ be the set of axis-parallel rectangles in the plane. A rectangle $R \in \mathbf{R}$ is characterized by two points: the lower left corner $ll(R)$ and the upper right corner $ur(R)$. If $ll(R)$ and $ur(R)$ differ in both x- and y-coordinate, we call R *non-trivial*. Since R is axis-parallel, the upper left corner $ul(R) = \big(ll(R)_x, ur(R)_y\big)$ and the lower right corner $lr(R) = \big(ur(R)_x, ll(R)_y\big)$ are uniquely defined. The *boundary* of R is given by the four line segments $l(R) = \big(ll(R), ul(R)\big)$ (left boundary), $u(R) = \big(ul(R), ur(R)\big)$ (top boundary), $r(R) = \big(ur(R), lr(R)\big)$, (right boundary), $b(R) = \big(lr(R), ll(R)\big)$ (bottom boundary). The *open intersection* $S = Q \cap R$ of two rectangles Q and R is non-empty if and only if S is a non-trivial rectangle, *i.e.*, both width and height of S must be strictly greater than zero.

We concentrate on the six different *labeling models* in Figure 1. The *discrete* models or *fixed-position models* displayed in Figure 1(a)-(c) allow only a finite number of feasible positions per label. The more general *slider models* (Figure 1(d)-(f)) allow a continuous movement of a label around the appropriate point feature. We formally state the problem as follows:

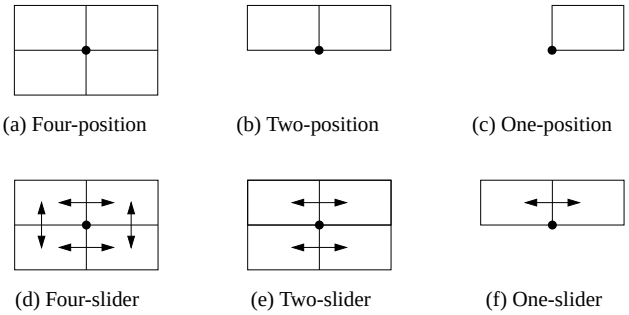
Problem 1.1 (Label Number Maximization, LNM). Given a set Λ (the *labels*), two functions $w, h : \Lambda \to \mathbb{Q}$ (the widths and heights of the labels), and a function $a : \Lambda \to \mathbb{Q} \times \mathbb{Q}$ (the points to be labeled), find a subset $\Lambda_P \subseteq \Lambda$ of largest cardinality and a function $\rho : \Lambda_P \to \mathbf{R}$, so that the following conditions hold:

(L1) Rectangle $\rho(\lambda)$ has width $w(\lambda)$ and height $h(\lambda)$ for every $\lambda \in \Lambda_P$.
(L2) Function ρ places all rectangles according to the *labeling model*, that is,

$$a(\lambda) \in \begin{cases} l(\rho(\lambda)) \cup u(\rho(\lambda)) \cup r(\rho(\lambda)) \cup b(\rho(\lambda)) & \text{in the } \textit{four-slider model} \\ t(\rho(\lambda)) \cup b(\rho(\lambda)) & \text{in the } \textit{two-slider model} \\ b(\rho(\lambda)) & \text{in the } \textit{one-slider model} \\ \{ll(\rho(\lambda)), ul(\rho(\lambda)), ur(\rho(\lambda)), lr(\rho(\lambda))\} & \text{in the } \textit{four-position model} \\ \{ll(\rho(\lambda)), lr(\rho(\lambda))\} & \text{in the } \textit{two-position model} \\ \{ll(\rho(\lambda))\} & \text{in the } \textit{one-position model} \end{cases}$$

for every $\lambda \in \Lambda_P$.
(L3) The open intersection $\rho(\lambda) \cap \rho(\mu)$ is empty for all distinct $\lambda, \mu \in \Lambda_P$.

We call an assignment of labels to rectangles that satisfies the three properties (L1)–(L3) a *labeling*. Properties (L1) and (L2) make sure that each label λ is attached correctly to its point feature $a(\lambda)$ and drawn with the given size. Property (L3) forbids overlaps between the labels. We allow, however, that two labels touch each other. The label number maximization problem LNM is the problem of finding a labeling of maximum size. The *decision problem* asks the question whether there exists a labeling for $\Lambda_P = \Lambda$. In the following, let n denote the number of labels.



(a) Four-position (b) Two-position (c) One-position

(d) Four-slider (e) Two-slider (f) One-slider

Fig. 1. Axis-parallel rectangular labeling models. A label can be placed in any of the positions indicated by the rectangles and slid in the directions of the arcs

1.1. Discrete Models

Most previous work on map labeling concentrates on the discrete models, the most popular of which is the four-position model (see Figure 1). The two- and one-position models have been introduced rather for theoretical purposes. While the decision problem in the four- and the p-position model (for fixed $p \geq 4$) is *NP*-complete (see Kato and Imai, 1988; Formann and Wagner, 1991; Marks and Shieber, 1991), it can be solved in time $\Theta(n \log n)$ via a 2-satisfiability formulation in the two-position model (Formann and Wagner, 1991), and via a simple sweep-line algorithm in the one-position model. Iturriaga (1999) shows *NP*-completeness in the 3-position model.

These results can directly be extended to the label size maximization problem; it is polynomially solvable in the one- and two-position model and *NP*-hard in the p-position model for fixed $p \geq 4$. For the label size maximization problem in the four-position model with unit squares, Formann and Wagner (1991) present a $\frac{1}{2}$-approximation algorithm running in $O(n \log n)$ time. Moreover, they show that no better approximation factor is possible unless *P* is equal to *NP*. Kučera, Mehlhorn, Preis, and Schwarzenecker (1993) present an $O(4^{\sqrt{n}})$ time algorithm that is able to solve instances with up to 100 labels to provable optimality.

In contrast to the label size maximization problem, the label number maximization problem in the one- and two-position model is *NP*-hard. Woeginger (2000) shows *NP*-hardness of the two problems by a reduction from the maximum independent set problem in planar cubic graphs that has been shown to be *NP*-hard in (Garey, Johnson, and Stockmeyer, 1976). *NP*-hardness for LNM in the p-position models follows directly from the *NP*-completeness result for the corresponding decision problems.

Agarwal, van Kreveld, and Suri (1998) study the label number maximization problem (LNM) in the p-position model for fixed p. They present a $\frac{1}{2}$-approximation algorithm for labels of unit height. The $\Theta(n \log n)$-time algorithm divides the problem into one-dimensional subproblems for which it computes the largest non-overlapping set of intervals with a greedy strategy. The authors also present a polynomial-time approximation scheme in the p-position model when the rectangles have unit height. The algorithm finds a solution of size at least $|\Lambda_P|/(1 + \frac{1}{k})$ for $k \geq 1$ in time $O(n \log n + n^{2k-1})$, where Λ_P is an optimal solution. If the rectangles differ in their height, the authors

develop a factor $O(\log n)$-approximation algorithm running in time $\Theta(n \log n)$. So far, for this problem no constant factor approximation algorithm is known.

The first exact algorithms for the label number maximization problem (LNM) in the p-position model are proposed by Cromley (1986) and Zoraster (1990). Both authors experiment with a zero-one integer linear programming formulation for LNM that they solve approximately using Lagrangian relaxation, subgradient optimization techniques, and several problem-specific heuristics. However, they can not solve practically relevant instances to provable optimality with their approach.

Verweij and Aardal (1999) present the only practically efficient algorithm for computing provably optimal solutions for LNM in the discrete models. 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 (Christensen, Marks, and Shieber, 1995)) within moderate computation time (about 20 minutes) and up to 950 point features within two hours.

Many articles have been published on heuristic algorithms for the label number maximization problem and the label size maximization problem (see Wolff and Strijk). Here, we will only mention the paper by Christensen *et al.* (1995). The authors present a comprehensive treatment of the label number maximization problem (LNM) in the four-position model including a comparison of heuristic methods and an extensive computational study. Their procedure for randomly creating labeling instances has become a widely used benchmark generator in the map labeling literature.

1.2. Slider Models

More natural than the discrete models are the slider models that allow a continuous movement of a label around its point feature. Although Hirsch considered this model already in 1982, it was not further investigated until very recently.

In (van Kreveld, Strijk, and Wolff, 1999), the authors define several variations of slider models, namely, the one-, two-, and four-slider model (see Figure 1). *NP*-completeness of the decision problem in the four-slider model is shown independently by van Kreveld *et al.* (1999) and Marks and Shieber (1991), and in the two-slider model by Iturriaga and Lubiw (1997). The decision problem in the one-slider model can be solved using a simple greedy sweep-line algorithm whereas the corresponding label number maximization problem is *NP*-hard (Woeginger, 2000)[1].

We give an overview of the complexity for the decision problem and the label number maximization problem in Table 1. In the cases in which the pure labeling problem is *NP*-complete, the label number maximization problem is obviously *NP*-hard. It is interesting that all maximization versions—even in the seemingly simple one-position model—are *NP*-hard.

Van Kreveld *et al.* (1999) present a $\frac{1}{2}$-approximation algorithm that is able to find a solution of LNM in any of the slider-models with unit height rectangles. The algorithm is a $\Theta(n \log n)$ greedy sweep-line algorithm. The sweep-line proceeds while repeatedly choosing the label whose right edge is leftmost among all remaining label candidates

[1] using a similar construction as in his proofs for the discrete models

Model	Decision problem	Label number max. problem
One-position	$O(n \log n)$ Simple plane sweep	*NP*-hard (Woeginger, 2000)
Two-Position	$O(n \log n)$ (Formann and Wagner, 1991)	*NP*-hard (Woeginger, 2000)
Four-Position	*NP*-complete (Kato and Imai, 1988; Marks and Shieber, 1991; Formann and Wagner, 1991)	*NP*-hard
One-Slider	$O(n \log n)$ Greedy sweep-line, (Woeginger, 2000)	*NP*-hard (Woeginger, 2000)
Two-Slider	*NP*-complete (Iturriaga and Lubiw, 1997)	*NP*-hard
Four-Slider	*NP*-complete (van Kreveld *et al.*, 1999; Marks and Shieber, 1991)	*NP*-hard

Table 1. A complexity overview of the decision and the number maximization problem for point feature labeling in different models (n is the number of labels)

if possible. For the same models, the authors develop a polynomial time approximation scheme. The respective algorithms label at least $(1 - \epsilon)$ times the optimum number in overall running time $O(n^{4/\epsilon^2})$.

Strijk and van Kreveld (1999) extend the above mentioned $\frac{1}{2}$-approximation algorithm for the slider models to labels with different heights. If r denotes the number of different label heights, the running time of the algorithm is $O(rn \log n)$.

An interesting point in (van Kreveld *et al.*, 1999) is the investigation of the relationship between the discrete and the slider models shown in Figure 1. Given unit square label candidates in a model M_1 and another model M_2, the factor Ψ_{M_1,M_2} denotes how many more points can receive labels under the model M_1 compared to that of model M_2. The authors show that Ψ_{M_1,M_2} equals two for many of the relationships. It is worth noting that the factor of the four-slider model compared to the four-position model is between two and four, whereas the factor of the four-slider model compared to the one-slider model is between two and three. Also the computational results on the six models confirm the theoretical result that the slider models are significantly better than the discrete models. The four-slider model allows to place up to 15% more labels in real-world instances and up to 92% more labels in pseudo-random instances as compared to the four-position model.

1.3. A New Approach Using Constraint Graphs

We will present an algorithm for the label number maximization problem that works in any of the above mentioned labeling models. We allow several labels per point feature and labels of different sizes. Furthermore, our method easily generalizes to combinations and variations of labeling models, such as horizontally sliding labels for some points and vertically sliding labels for the other points or discrete models where for different points the possible positions differ. Figures 2 and 3 show provably optimal labelings of a practical instance in the one-position and in the four-slider model computed with an implementation of our algorithm.

Fig. 2. Optimal solution of a map with German railway stations in the one-position model. 235 of 366 cities have labels with text font size 8pt (as compared to 294 and 339 in the two- and four-position model, respectively). Unlabeled cities are drawn as bigger circles

The new approach is based on a pair of *constraint graphs*. Informally, the directed edges of a constraint graph represent precedence constraints between the objects corresponding to their endpoints. These relations may additionally be specified by a weight on the arcs. Constraint graphs have been used in the areas of scheduling, VLSI design and graph drawing.

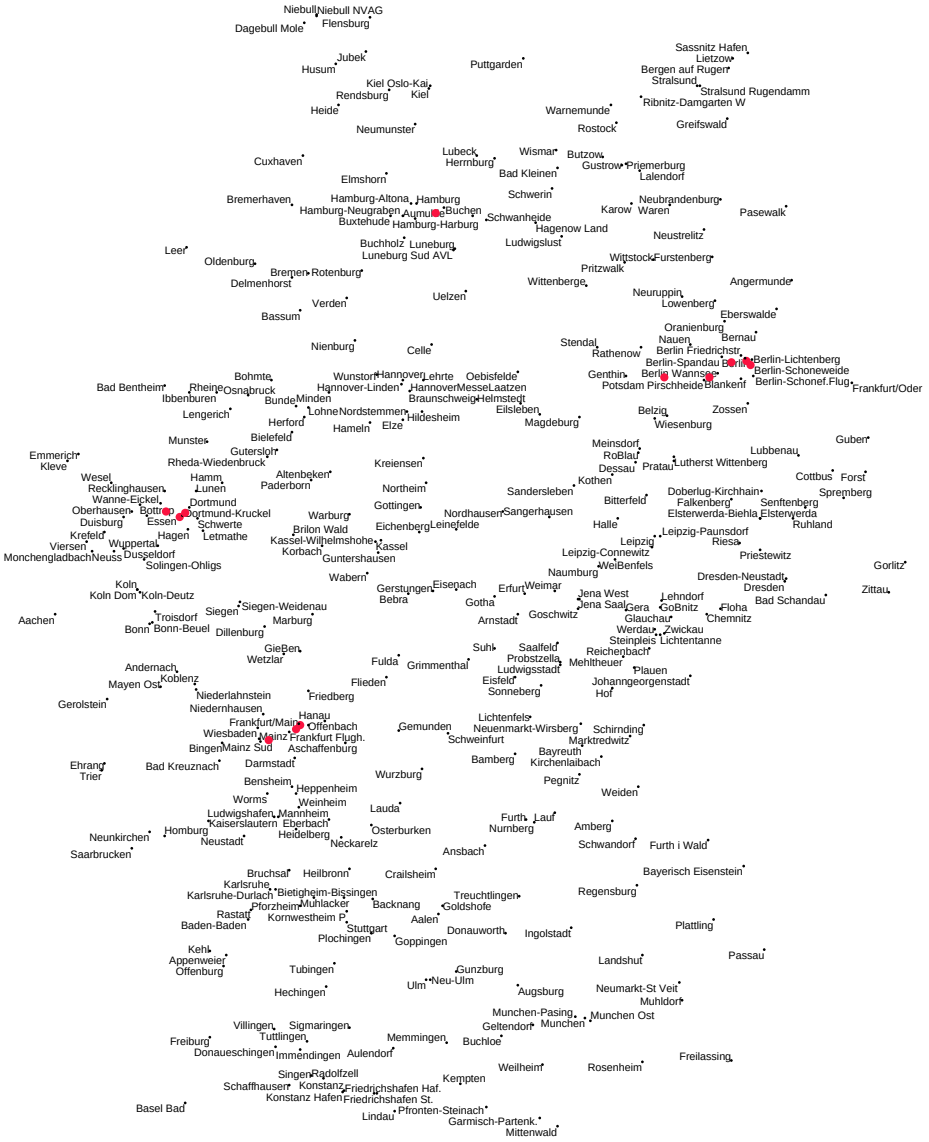


Fig. 3. Optimal solution of the German railway station instance in the four-slider model. 354 cities are labeled (as compared to 311 and 349 in the one- and two-slider models, respectively). See also Figure 2

Our key idea is to decompose the label number maximization problem into a horizontal and vertical problem component and link the two components by additional constraints. For each of the two components we construct a constraint graph that is based on the information in the instance of the labeling problem. During the construction of the graphs we determine a set of potential additional arcs and a set of constraints that must be satisfied in a feasible solution. We give a combinatorial formulation for the

label number maximization problem and define the maximum constraint graph satisfaction problem (MCGS). We prove that the combinatorial problem MCGS is equivalent to the label number maximization problem LNM.

A related characterization is used by Bartusch, Möhring, and Radermacher (1988) in the area of scheduling. In scheduling, the nodes of a constraint graph correspond to the jobs and the weighted edges characterize the temporal constraints between these jobs. The authors consider optimization in the set of feasible schedules with resource constraints and time windows. They characterize this set as extensions of a given partial order that satisfy certain order-theoretic properties. Because of the one-dimensional nature of their problem they do not have to consider the interactions between different constraint graphs.

We present a zero-one integer linear programming formulation for MCGS and prove *NP*-hardness of the separation problem for the class of positive cycle inequalities, *i.e.*, the problem of finding a violated inequality of this class given a fractional LP-solution of the relaxation. We show that we can use a feasible solution of the integer linear program to construct a feasible labeling. In particular, this enables us to find an optimal labeling. We present an iterative branch-and-bound scheme that is based on the pure zero-one formulation. Compared to alternative techniques to solve the ILP developed in this paper, the iterative scheme has the advantage that it avoids the *NP*-hard separation problem. We find it remarkable that our approach is independent of the labeling model and results in a discrete algorithm even if the problem is of continuous nature as in the slider models.

This paper is organized as follows: In Section 2 we describe the construction of the pair of constraint graphs and state the maximum constraint graph satisfaction problem (MCGS). We prove that problem MCGS is equivalent to the label number maximization problem.

Section 3 contains the integer linear programming formulation for the maximum constraint graph satisfaction problem, the *NP*-hardness proof of the separation problem, and a description of the new algorithm.

We describe extensive computational experiments with the new approach in Section 4 and test our implementation on publicly available benchmark data that include random instances as well as real-world data. Additionally we run our algorithm on instances created by the benchmark generator presented in (Christensen *et al.*, 1995).

Note that our formulation of LNM allows labels to overlap unlabeled point features (which is desirable in some applications). We describe in Section 6 how we can exclude these overlaps and investigate whether our algorithm can satisfy additional criteria like preferable positions and labels of different importance. We conclude and mention directions for further research in Section 6.

2. LNM as a Combinatorial Optimization Problem

In this section we reformulate the label number maximization problem LNM as a problem of a combinatorial nature. We introduce the concept of *constraint graphs* and show how to construct these graphs. We link the two graphs by additional constraints and state

the maximum constraint graph satisfaction problem MCGS. At the end of this section we show that MCGS is equivalent to LNM.

2.1. Constraint Graphs

An important feature of the label number maximization problem LNM is its possible decomposition into a horizontal and a vertical problem component; this observation motivates us to treat each direction separately. We have already used this technique for two problems in graph drawing: The two-dimensional compaction problem and a combined compaction and labeling problem (Klau and Mutzel, 1999b,a).

Nodes in the *horizontal constraint graph* $D_x = (V_x, A_x)$ represent x-coordinates of objects in the problem, weighted directed edges represent horizontal distance relations between the objects that correspond to their endpoints. Similarly, the directed graph D_y codes the vertical relationships.

Definition 2.1. *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 that respects the arcs of this pair to construct a feasible labeling—as long as the graphs satisfy certain conditions. The following theorem expresses an important connection between constraint graphs and coordinate assignments. Its proof follows, *e.g.*, from (Cormen, Leiserson, and Rivest, 1990, Theorem 25.17).

Theorem 2.1. *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.*

In the following, we describe the construction of the pair (D_x, D_y) for a given instance of the label number maximization problem.

2.2. Modeling 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 4 shows a set of point features and its representation in the constraint graphs.

Observation 1. *A coordinate assignment* c *that respects* A_F *results in a correct placement of point features (up to translation).*

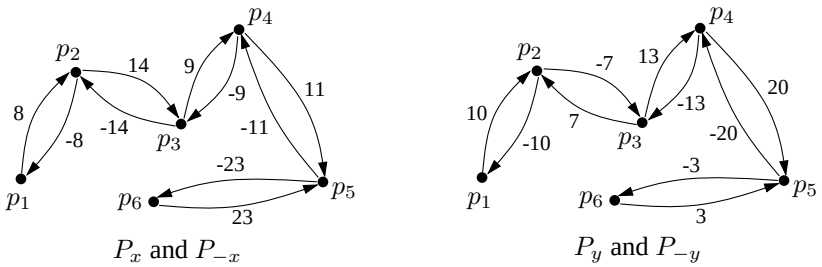


Fig. 4. Modeling the placement of point features with fixed distance arcs

2.3. Modeling Labels

Each label $\lambda \in \Lambda$ has to be represented by a rectangle $\rho(\lambda)$ of width $w(\lambda)$ and height $h(\lambda)$. Additionally, we have to ensure that λ will be placed correctly with respect to the labeling model.

Straightforwardly, we model a label λ by two nodes in V_x and two nodes in V_y, representing the coordinates of $\rho(\lambda)$. 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* $A_L(\lambda) = \{(l_\lambda, r_\lambda), (r_\lambda, l_\lambda), (b_\lambda, t_\lambda), (t_\lambda, b_\lambda)\}$ in order to model the size of $\rho(\lambda)$. 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 Figure 5(a).

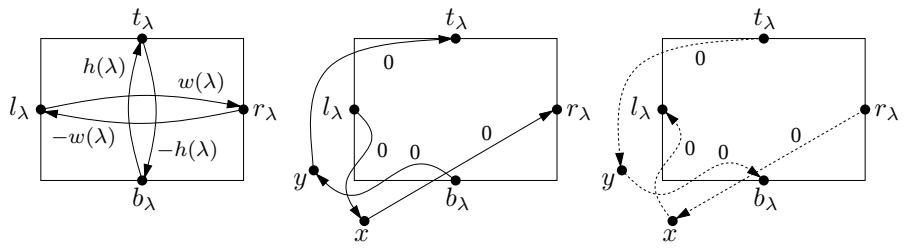
A label λ must be placed close to its point feature $a(\lambda)$. Let x and y be the nodes representing point $a(\lambda)$ in the constraint graphs. We add four *proximity arcs* $A_P(\lambda) = \{(x, r_\lambda), (l_\lambda, x), (y, t_\lambda), (b_\lambda, y)\}$ as illustrated in Figure 5(b). These arcs have zero weight and exclude that the point feature $a(\lambda)$ lies outside the rectangle $\rho(\lambda)$.

The point feature may still lie inside $\rho(\lambda)$. We can 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. As it will turn out we have to be careful which boundary arcs to choose and we defer a detailed characterization to the definition of the Maximum Constraint Graph Satisfaction problem on page 12. 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 $\rho(\lambda)$ to be equal to the coordinate of x; the label has to be placed at its leftmost position. See also Figure 5(c).

At this point we can influence the labeling model and define the boundary arcs as follows:

$$A_B(\lambda) = \begin{cases} \{(r_\lambda, p_x), (p_x, l_\lambda), (t_\lambda, p_y), (p_y, b_\lambda)\} & \text{in the 4-slider or 4-position model} \\ \{(r_\lambda, p_x), (p_x, l_\lambda), (p_y, b_\lambda)\} & \text{in the 2-position model} \\ \{(t_\lambda, p_y), (p_y, b_\lambda)\} & \text{in the 2-slider model} \\ \{(p_x, l_\lambda), (p_y, b_\lambda)\} & \text{in the 1-position model} \\ \{(p_y, b_\lambda)\} & \text{in the 1-slider model} \end{cases}$$

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 $A_B(\lambda)$



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

Fig. 5. Modeling a label λ

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 labeling models we have introduced in Section 1 as additional requirements on the constraint graphs.

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

2.4. Avoiding Label Overlaps

Until now we have assured that each label is placed correctly with respect to its point feature. It remains to 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.

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

Let R_λ be 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 $A_S(\lambda, \mu) = \emptyset$.

Consider now the case that the intersection of R_λ and R_μ is not empty, as depicted in Figure 6. 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 where $A_S(\lambda, \mu)$ contains all four label separation arcs occurs if λ and μ label the same point feature, *i.e.*, $a(\lambda) = a(\mu)$.

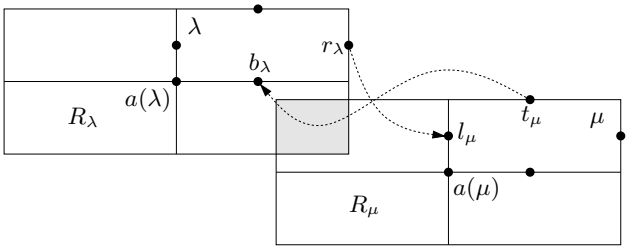


Fig. 6. Label separation arcs between two labels λ and μ

Observation 3. *Let λ and μ be two labels that can possibly overlap and let c be a coordinate assignment respecting $A_L(\lambda)$, $A_L(\mu)$ and $A \subseteq A_S(\lambda, \mu)$ with $|A| \geq 1$. Then c results in a placement where the two rectangles $\rho(\lambda)$ and $\rho(\mu)$ do not overlap.*

2.5. Satisfying the Constraint Graphs

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

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

and state the label number maximization 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 2.1 (Maximum Constraint Graph Satisfaction, MCGS). 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 models and let A_x and A_y be the arc sets of D_x and D_y, respectively. Find a set $\Lambda_P \subseteq \Lambda$ of greatest cardinality and an arc set $A \subseteq A_{\text{pot}}$ with the properties:

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

Theorem 2.2. *Problems* LNM *and* MCGS *are polynomially equivalent.*

Proof. Let A be a solution of MCGS. We extend (D_x, D_y) by adding A to the arc sets. Because of property (F3) and Theorem 2.1, there is a coordinate assignment that respects both the horizontal and vertical arc set. Observations 1, 2, and 3 ensure that properties (L1), (L2) and (L3) are satisfied, 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 $\rho(\lambda)$ 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 labeling. 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 Theorem 2.1. □

3. Integer Linear Programming Formulation

We have shown how to transform the label number maximization problem into the combinatorial optimization problem MCGS. In this section we present an integer linear programming formulation to solve problem MCGS to provable optimality. We first develop the zero-one formulation and then show that the separation problem of finding violated positive cycle inequalities is *NP*-hard. We want to mention that we have also developed an alternative, extended integer linear programming formulation that avoids the cycle inequalities at the cost of additional variables and constraints. See the thesis (Klau, 2001) for details. We conclude the section by describing an iterative branch-and-bound scheme that is based on the zero-one formulation.

3.1. Zero-One Formulation

In order to solve problem MCGS we have 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 or not. Additionally, there are variables $x_a \in \{0, 1\}$ for potential additional arcs $a \in A_{\text{pot}}$. We define

$$y_\lambda = \begin{cases} 1 & \lambda \in \Lambda_P \\ 0 & \lambda \in \Lambda \setminus \Lambda_P \end{cases}$$

and

$$x_a = \begin{cases} 1 & a \in A \\ 0 & a \in A_{\text{pot}} \setminus A \end{cases} .$$

We present the zero-one integer linear program (ILP) and show that it corresponds to MCGS. We refer to (ILP.1) as *boundary inequalities*, to (ILP.2) as *label separation inequalities* and to (ILP.3) as *positive cycle inequalities* since they correspond to the appropriate properties in the combinatorial problem on page 12.[2]

[2] Based on the formulation (ILP) we have developed an integer linear program in which the y-variables are substituted by changing boundary inequalities to equalities and adding one class of inequalities, see (Klau, 2001) for details. In practice however, this formulation has turned out to be inferior.

$$\max \quad \sum_{\lambda \in \Lambda} y_\lambda \qquad \text{(ILP)}$$

$$\text{subject to} \quad \sum_{a \in A_B(\lambda)} x_a \geq d \cdot y_\lambda \qquad \forall \lambda \in \Lambda \qquad \text{(ILP.1)}$$

$$\sum_{a \in A_S(\lambda,\mu)} x_a - y_\lambda - y_\mu \geq -1 \qquad \forall \lambda, \mu \in \Lambda, \lambda \neq \mu \qquad \text{(ILP.2)}$$

$$\sum_{a \in C \cap A_{\text{pot}}} x_a \leq |C \cap A_{\text{pot}}| - 1 \qquad \forall \text{ positive cycles } C \qquad \text{(ILP.3)}$$

$$y_\lambda \in \{0,1\} \qquad \forall \lambda \in \Lambda \qquad \text{(ILP.4)}$$

$$x_a \in \{0,1\} \qquad \forall a \in A_{\text{pot}} \qquad \text{(ILP.5)}$$

Theorem 3.1. *Each feasible solution (y, x) to (ILP) corresponds to a feasible solution of* MCGS *and vice versa. The value of the objective function is equal to $|\Lambda_P|$.*

Proof. Consider a feasible solution (y, x) to the ILP. Each entry in (y, x) is zero or one because of the integrality conditions (ILP.4) and (ILP.5). We set $\Lambda_P = \{\lambda \in \Lambda \mid y_\lambda = 1\}$ and $A = \{a \in A_{\text{pot}} \mid x_a = 1\}$. Clearly, $|\Lambda_P| = \sum_{\lambda \in \Lambda} y_\lambda$. For each $\lambda \in \Lambda_P$ inequality (ILP.1) turns to $\sum_{a \in A_B(\lambda)} x_a \geq d$, it follows property (F1). Likewise, (ILP.2) yields $\sum_{a \in A_S(\lambda,\mu)} x_a \geq 1$ if both λ and μ are in Λ_P, satisfying (F2). Obviously, inequalities (ILP.3) ensure property (F3).

For the other direction, consider a feasible solution of MCGS. We set y_λ to one if $\lambda \in \Lambda_P$ and to zero otherwise. In the same manner, we set x_a to one if $a \in A$, otherwise to zero. Similar arguments as above show that (y, x) does not violate any of the inequalities, thus it is a feasible solution for (ILP). □

Corollary 3.1. *An optimal solution of (ILP) corresponds to an optimal labeling.*

Corollary 3.1 and Theorem 2.1 suggest an algorithm for attacking practical instances of the label number maximization problem: In a first step, we solve the ILP 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.

Remark 3.1. Although Theorem 2.1 is existential, it can be proved constructively: it is easy to find an assignment for a constraint graph without positive cycles, *e.g.* by computing minimum-cost flows.

A crucial issue in solving our integer linear programming formulation (ILP) for the label number maximization problem concerns finding violated positive cycle inequalities. As we will show next, this separation problem is *NP*-hard.

3.2. Complexity of Finding Positive Cycles

The decision version of the corresponding separation problem is the following problem:

Problem 3.1. Given a digraph $D = (V, A)$, a weight vector $d \in \mathbb{Z}^{|A|}$, and a fractional LP-solution $\chi \in [0..1]^{|A|}$. Does there exist a directed cycle $C \subseteq A$ with $\sum_{a \in C} \chi_a > |C| - 1$ and $\sum_{a \in C} d_a > 0$?

We set $\xi_a = 1 - \chi_a$ for all $a \in A$ and obtain an equivalent version of Problem 3.1.

Problem 3.2 (Positive Weight-Constrained Cycle, PWCC).
Given a digraph $D = (V, A)$, a weight vector $d \in \mathbb{Z}^{|A|}$, and a vector $\xi \in [0..1]^{|A|}$. Does there exist a directed cycle $C \subseteq A$ with $\sum_{a \in C} \xi_a < 1$ and $\sum_{a \in C} d_a > 0$?

The following problem is *NP*-complete (see Garey and Johnson, 1979, [ND30]):

Problem 3.3 (Shortest Weight-Constrained Directed Path, SWCP). Given a digraph $D = (V, A)$, a length function $l : A \to \mathbb{N}$, a weight function $w : A \to \mathbb{N}$, two specified vertices $s, t \in V$, and integers W and K. Does there exist a simple directed path in D from s to t with total weight W or less and total length K or less?

Theorem 3.2. *The positive weight-constrained cycle problem is NP-complete in the weak sense.*

Proof. The problem is in *NP* because a nondeterministic algorithm need only guess a directed cycle C in D and check in polynomial time whether its weight is positive and whether $\xi(C) < 1$. Let $I_{\text{SWCP}} = \big((V, A), l, w, s, t, W, K\big)$ be an instance for the shortest weight-constrained directed path problem. We transform it into an instance $I_{\text{PWCC}} = (D', d, \xi)$ of the positive weight-constrained cycle problem.

Let $L = \max\{l(a) \mid a \in A\}$. We set $D' = (V, A')$ with $A' = A \cup \{(t, s)\}$) and define the two vectors as follows:

$$d_a = \begin{cases} -w(a) & a \in A \\ W + 1 & a = (t, s) \end{cases} \qquad \xi_a = \begin{cases} \frac{l(a)}{L+K+1} & a \in A \\ 1 - \frac{K+1}{L+K+1} & a = (t, s) \end{cases} .$$

Obviously, we have $d \in \mathbb{Z}^{|A'|}$ and $0 \le \xi_a \le 1$ for $a \in A'$.

We claim that there is a simple path from s to t in D with length at most K and weight at most W if and only if there is a directed cycle $C \subseteq A'$ with $\sum_{a \in C} d_a > 0$ and $\xi(C) < 1$.

Let p be such a path in D and let C be the cycle that results from appending the arc (t, s) to p. Cycle C in D' has weight

$$\begin{aligned} \sum_{a \in C} d_a &= \sum_{a \in p} -w(a) + d_{(t,s)} = -\sum_{a \in p} w(a) + W + 1 \\ &\ge -W + W + 1 = 1 > 0 \end{aligned}$$

$$\text{and } \xi\text{-value } \sum_{a\in C}\xi_a = \sum_{a\in p}\left(\frac{l(a)}{L+K+1}\right) + 1 - \frac{K+1}{L+K+1}$$
$$= \frac{1}{L+K+1}\underbrace{\sum_{a\in p} l(a)}_{\leq K} + \frac{L}{L+K+1}$$
$$\leq \frac{K}{L+K+1} + \frac{L}{L+K+1} = \frac{L+K}{L+K+1} < 1 \ .$$

For the backward direction of the proof let C be a cycle in D' with positive weight and $\xi(C) < 1$. Note that the arc (t, s) must lie on C, because it is the only arc with positive weight d. It follows that the remaining arcs of C form a path p from s to t in D. The w-weight of p is

$$\sum_{a\in p} w(a) = \sum_{a\in p} -d_a = W + 1 - \sum_{a\in p} d_a - W - 1$$
$$= W + 1 - \underbrace{\sum_{a\in C} d_a}_{\geq 1} \leq W \ .$$

$$\text{The length of } p \text{ is } \sum_{a\in p} l(a) = \sum_{a\in p}(L+K+1)\xi_a$$
$$= (L+K+1)\sum_{a\in p}\xi_a + (L+K+1)(1 - \frac{K+1}{L+K+1})$$
$$-(L+K+1)(1 - \frac{K+1}{L+K+1})$$
$$= (L+K+1)\underbrace{\sum_{a\in C}\xi_a}_{<1} -(L+K+1) + K + 1$$
$$< L+K+1-(L+K+1)+K+1 = K+1 \ .$$

Because all lengths are integral, we have $l(p) \leq K$.

The Shortest Weight-Constrained Path problem admits a fully polynomial time approximation scheme as shown by Hassin (1992) and Philips (1993). This result is directly transferable to the Positive Weight-Constrained Cycle problem by plugging it into the first part of the above proof. Thus, the Positive Weight-Constrained Cycle problem is *NP*-complete only in the weak sense. □

3.3. Implementation

We have implemented an iterative branch-and-bound approach to solve the label number maximization problem. For the underlying constraint graphs we use LEDA, a library of efficient data structures and algorithms (Mehlhorn and Näher, 1999). We use

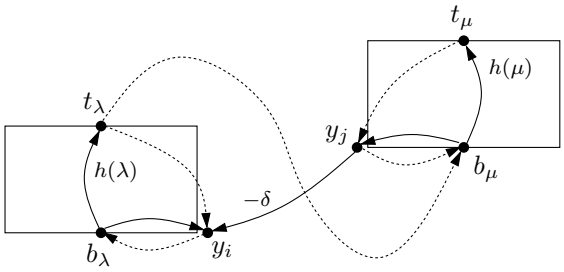


Fig. 7. Local positive cycles in the vertical constraint graph D_y arising from the interaction of labels. Potential arcs are dotted, $\delta = y(p_j) - y(p_i)$

the optimization library (CPLEX) to solve the ILPs that occur in the iterative scheme. As described in Section 2.3, we determine the labeling model by changing the rules for constructing the set of boundary arcs.

A common preprocessing phase partitions a given instance of the label number maximization problem in several components. Let G_L be the graph that contains a vertex for each label and an edge for each pair of labels λ and μ with non-empty intersection $R_\lambda \cap R_\mu$, *i.e.*, for pairs of labels that possibly overlap. It is obvious that we can process the connected components of G_L separately.

The initial set of inequalities consists of the boundary inequalities, the label separation inequalities and a subset of positive cycle inequalities. We call these inequalities *local positive cycle inequalities* and we can determine them in advance by looking at possible positive cycles involving up to two labels:

Figure 7 illustrates the local cycles in the vertical case: The first type of cycles consists of two boundary arcs and one label size arc—in Figure 7, these are $\big((y_i, b_\lambda)$, (b_λ, t_λ), $(t_\lambda, y_i)\big)$ and $\big((y_j, b_\mu), (b_\mu, t_\mu), (t_\mu, y_j)\big)$. We exclude these cycles by adding the inequalities $x_{(y_i,b_\lambda)} + x_{(t_\lambda,y_i)} \leq 1$ and $x_{(y_j,b_\mu)} + x_{(t_\mu,y_j)} \leq 1$. Depending on the arc weights, a second type of positive cycle may appear that involves two labels linked by a label separation arc. If, as in Figure 7, the height of λ is greater than the vertical distance between p_i and p_j, the cycle $\big((b_\lambda, t_\lambda)$, (t_λ, b_μ), (b_μ, y_j), (y_j, y_i), $(y_i, b_\lambda)\big)$ has positive weight. In this case, we add the inequality $x_{(t_\lambda,b_\mu)} + x_{(y_i,b_\lambda)} \leq 1$. A similar situation, $h(\mu) > y(p_j) - y(p_i)$, results in inequality $x_{(t_\lambda,b_\mu)} + x_{(t_\mu,y_j)} \leq 1$. If the sum of both heights exceeds the vertical distance between the point features, *i.e.*, $h(\lambda) + h(\mu) > y(p_j) - y(p_i)$, we add the inequality $x_{(t_\lambda,b_\mu)} + x_{(t_\mu,y_j)} + x_{(y_i,b_\lambda)} \leq 2$.

The iterative branch-and-bound approach uses a commercial ILP-solver and works as follows: Let $\bar{x}$ be an optimal solution of an ILP in the iteration. If the corresponding constraint graphs do not contain positive cycles, $\bar{x}$ is an optimal solution for problem MCGS and thus for the label number maximization problem. Otherwise, we find one or more positive cycles, because the entries in $\bar{x}$ are not fractional, and—in this special case—we can solve the separation problem in polynomial time. We add the corresponding inequalities to our new integer linear program and iterate. In practice, we have found it advantageous not to compute optimal but just feasible solutions in the first iterations. We stop with the kth feasible solution ($k = \{1, 2, 4, 8, 16\}$) until we compute optimal solutions. Again, the details of this method can be found in (Klau, 2001).

The algorithm will terminate with an optimal solution for the maximum constraint graph satisfaction problem. We construct an optimal solution for the label number maximization problem as described in Corollary 3.1. The benefits of the iterative method lie in the fact that it avoids both the hardness of the separation problem and the additional variables, constraints, and constants of an extended formulation.

4. Computational Experiments

In this section we report our computational results for the label number maximization problem. All results in this section are computed by the iterative branch-and-bound scheme described in Section 3.3. We use a Sun Enterprise 450 with 1.1 GB main memory and two 400 MHz-CPUs for the experiments.

We test our integer linear programming-based approach on different types of instances in all six axis-parallel rectangular labeling models as defined in Figure 1 of the introduction.

Our first source of test data is publically available on the web page `http://www.math-inf.uni-greifswald.de/map-labeling/general/`. The site contains practical instances as well as randomly generated data. We test our implementation on several maps, *e.g.*, the German railway stations, or maps of the United States of America (see Figures 2, 3, and 8). The map of German railway stations contains 366 cities that have to be labeled with their names. We use a text font size of 8 pt. An optimal solution in the one-position model contains only 235 labels, see Figure 2. Already the two-position model allows a placement of 59 more labels (294). The algorithm places 311 labels in the one-slider model, 339 in the four-position model, and 349 in the two-slider model. Only 12 labels cannot be placed in an optimal solution for the four-slider model (354 labels, see Figure 3).

The running times show a remarkable difference of the performance of our implementation in the different labeling models. Whereas it is very fast in the one-position, two-position, and one-slider models (below 13 seconds), the computation takes about seven and a half minutes in the two-slider model and half an hour in the four-slider model. In the four-position model, the implementation needs more than two and a half days to find an optimal solution and prove its optimality. We defer a discussion of the reasons for the different running times to the end of this section.

For the example `us_abbrev_1041.xyn`, *i.e.*, the map of a major part of the United States of America, we scale the coordinates by a factor of 50 and use a text font size of 12 pt. Of the 1041 cities the exact algorithm labels 1004 in one hour in the four-position model (see Figure 8).

Additionally, we run our implementation on a set of practical data, the Munich drill hole instances from the above mentioned web page. The instances correspond to rectangular submaps of a map with 19,400 ground-water drill holes in the city of Munich with sizes n in the range $\{250, 500, 750, \ldots, 2750, 3000\}$. There are 30 instances of each size. The label sizes of these benchmark instances arise from heuristically scaling the labels to a large size for which still all the labels can be placed in the four-position model. Figure 9 shows such an instance with 500 drill holes. For these instances, we only compare the four-position and the four-slider model.

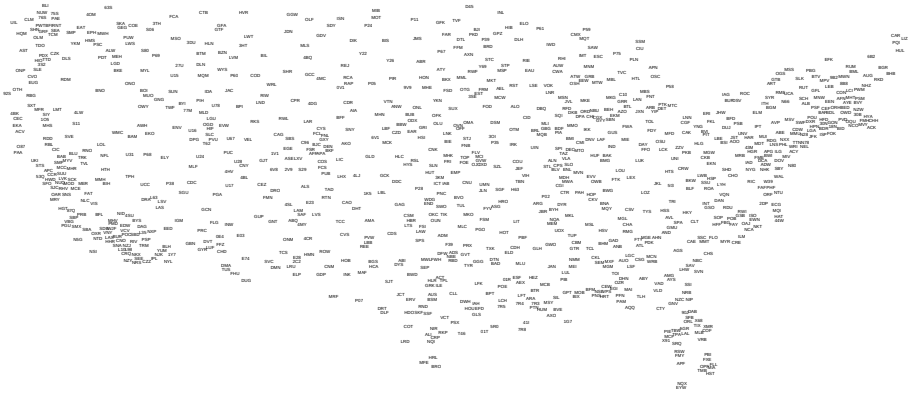


Fig. 8. Optimal labeling for a map of the U.S.A. without Alaska and Hawaii, 1004 of 1041 cities have labels. Instance `us_abbrev_1041.xyn`, coordinates multiplied by 50, font size 12 pt, four-position model

Fig. 9. 500 labeled ground-water drill holes in Munich

The plot in Figure 10 shows how long our implementation needs to place all the labels. It can be observed that the algorithm runs three to five times faster in the four-position model and is quite fast in both models. The good results indicate that our approach is suited to compute optimal solutions for large instances of the label size maximization problem.

We also test our implementation on a widely used benchmark generator for randomly creating instances of labeling problems, according to the rules described in (Christensen *et al.*, 1995): 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. Each

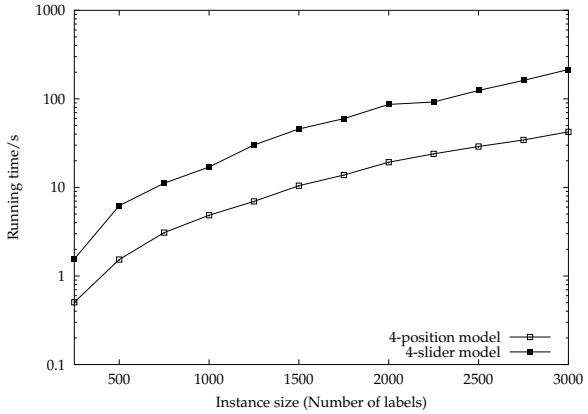


Fig. 10. Running times for the Munich drillholes instances (logarithmic scale)

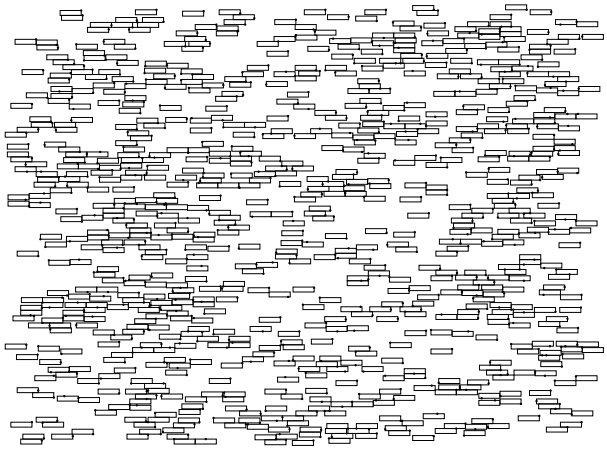


Fig. 11. A provable optimal solution of an instance with 800 point-features generated according to the rules in (Christensen *et al.*, 1995). 795 points receive labels, 40 minutes running time

point-feature has a label of width 30 and height 7. Figure 11 shows a provable optimal solution with 795 labels in the four-slider model for such an instance with $n = 800$.

Following the scheme in (Verweij and Aardal, 1999) we randomly generate 25 maps of size n with $n \in \{100, 150, \ldots, 950, 1000\}$. We limit the running time to 30 minutes of CPU-time for each component. Typically, only one or two difficult components exist, depending on the density of the instance. Table 2 shows for how many of the larger instances the computation of all components terminates within the time limit. Up to a size of 600 labels the implementation provides an optimal solution in all models in short computation time.

The one-position model is the only model for which our implementation produces optimal solutions for all 475 instances within the time limit. Looking at discrete and

	Number of solved instances/size								
Model	600	650	700	750	800	850	900	950	1000
1P	25	25	25	25	25	25	25	25	25
2P	25	25	25	25	22	21	15	5	-
4P	24	18	3	-	-	-	-	-	-
1S	25	25	25	25	25	25	24	19	12
2S	25	24	22	19	11	1	-	-	-
4S	25	23	20	13	3	1	-	-	-

Table 2. Number of optimally solved instances for the randomly generated data. The first row describes the size of the instances, the following rows show how many instances our implementation solves to optimality within 30 minutes CPU-time in the respective model

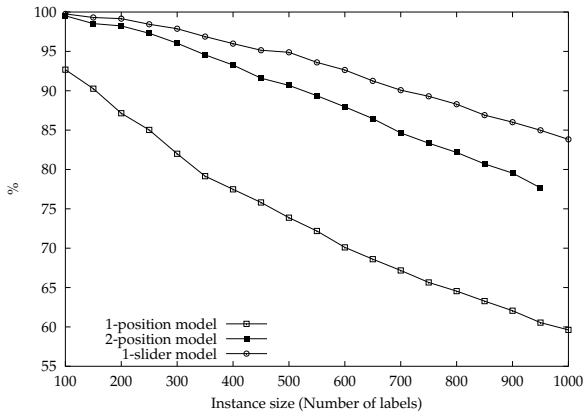


Fig. 12. Percentage of labels placed within each model: 1-position, 2-position, and 1-slider model

slider models independently, we observe that the more restricted a model is, the easier is the computation. In general, the discrete models are easier to solve with the exception of the four-position model: Some instances of the four-position model seem to be hardest for our implementation as already indicated by the railway station examples.

Figure 12 shows the percentage of labels that can be placed in each model for the "poor" models (one-position, two-position, and one-slider). We observe a linear decrease in the number of placed labels in relation to the total number of labels.

The same plot for the "good" models (four-position, two-slider, and four-slider) in Figure 13 shows a different behavior. In all models, the optimal solutions are quite close to the total number of labels, and their order reflects the freedom a label has in the respective model. However, the percentage of placed labels decreases quickly after an initial plateau which is close to 100%. Surprisingly, the trend of decreasing percentage seems to change for the larger instances. These anomalies in the four-slider and four-position model have the following reason: Only a few easy instances can be solved of that size, *i.e.*, instances which allow a relatively large number of placed labels.

Figure 14 shows the running times for the randomly generated instances in the discrete models with a logarithmic scale. After a certain instance size, the running times increase rapidly. For the one-position model, the threshold of the "exponential explo-

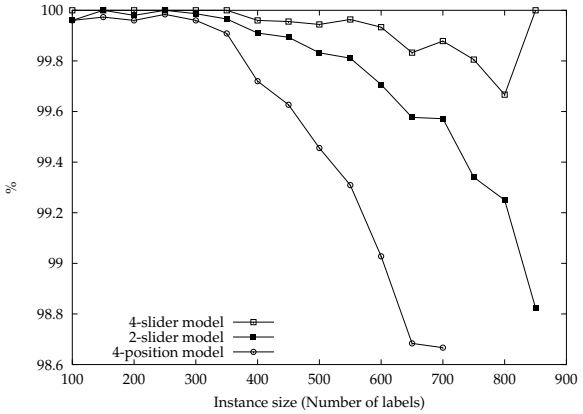


Fig. 13. Percentage of labels placed within each model: 4-position, 2-slider, and 4-slider model

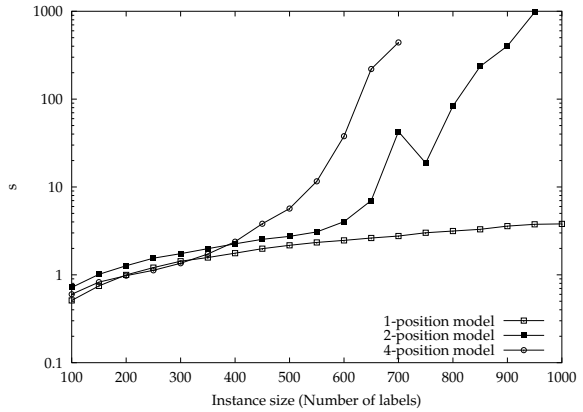


Fig. 14. Running times for the randomly generated instances, discrete models, logarithmic scale

sion" is at about 2,500 labels and is not reached by the sizes of the test data. Again, the anomalies in the plots are due to the fact that easy instances are also faster to solve.

For the slider models, the data look similar, see Figure 15. The bends in the plots exist for the same reason as for the discrete models. It can be seen that for the randomly generated data the four-slider and four-position model have about the same running time behavior.

We conclude the computational results by looking at the reasons why the performance of our approach depends such heavily on the labeling model. We can determine two main factors which influence the running time of our algorithm: On the one hand, this is the number of labels that cannot be placed in an optimal solution, *i.e.*, the difference $|\Lambda| - |\Lambda_P|$. In general, an optimal solution that allows a large number of placed labels is faster to compute since the number of possibilities for the set of removed labels is much smaller than in a solution in which many labels cannot be placed.

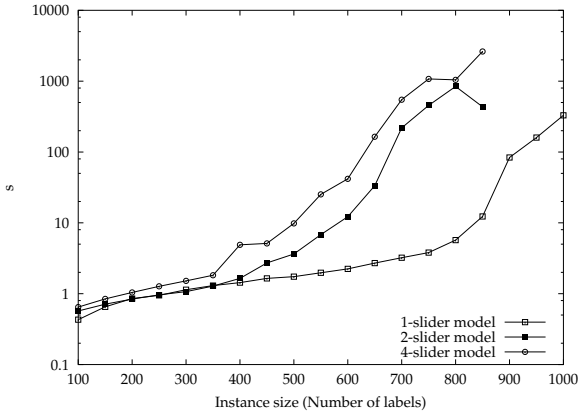


Fig. 15. Running times for the randomly generated instances, slider models, logarithmic scale

On the other hand, the tightness of the inequalities has an impact on the running time; the more restrictions on the variables, the faster the algorithms. Figures 14 and 15 reflect this order of tightness; the faster models are the more restricted ones.

Both factors, however, interrelate: In the more restricted models we can also place a smaller number of labels. This explains also the data in Table 2: in general, the inequalities are tighter for the discrete model, but in large instances the growing difference $|\Lambda| - |\Lambda_P|$ becomes more influential.

5. Extensions

As mentioned in Section 1, our formulation of LNM allows labels to overlap other, unlabeled point features. In several applications, this may be undesired. In this case, we take the point features into consideration when introducing the label separation arcs (which then should be called general separation arcs). We then have to check the overlap conditions also for point feature/label pairs; it is worth noting that then the boundary arcs arise as a special case of general separation arcs when considering a pair $(\lambda, a(\lambda))$, *i.e.*, a label and its point feature. A similar construction may be used to model obstacles that must not be touched by the labels.

In many applications, there are labels of different importance. It is easy to integrate this into our approach: The objective function of the integer linear program changes to $\sum_{\lambda\in\Lambda} z_\lambda y_\lambda$, where z_λ denotes the importance of label λ. The algorithm will then prefer more important labels and remove less important ones more easily. Another practically motivated extension is to model preferable positions of labels: Often, a label should be placed rather at its rightmost and upmost position than at other possible positions. We suggest to define a weight vector for the boundary arcs and incorporate it into the objective function.

6. Conclusions

We introduce a new approach for solving the label number maximization problem (LNM) for point features in six popular axis-parallel rectangular labeling models. One of the most important labeling models is the slider model, where labels can move continuously around point features. In our problem formulation we admit an arbitrary number of labels per point feature and different—but fixed—widths and heights for different labels.

We transform the labeling problem into the maximum constraint graph satisfaction problem (MCGS) which is a combinatorial problem in a pair of so-called *constraint graphs*. We show that the new formulation is equivalent to the original one. The key idea is to consider the horizontal and vertical problem component separately, linking them only by a small set of additional constraints.

We propose a zero-one integer linear programming formulation and present an iterative branch-and-bound scheme to solve instances of MCGS to optimality. To our knowledge this yields the first algorithms that compute provably optimal solutions in the slider models. Our extensive computational experiments show that our approach is practicable for real-world examples. We show how to extend our approach to satisfy practical requirements such as labels of different importance or preferable label positions. Currently, we use the constraint graph-based approach to solve large instances of the label size maximization problem.

Future work on this new approach to solve map labeling problems could exploit the relation to scheduling problems and d-dimensional orthogonal packing problems. Another interesting topic is the possible extension to line and area feature labeling.

Acknowledgements. The authors thank Gerhard Woeginger for filling three of the empty cells in Table 1 and improving a result in one of them, Matteo Fischetti for helpful discussions concerning the relation between the zero-one and the extended polytope, Alexander Wolff for the real-world data and the help with the conversion in our data format, Michael Jünger for spontaneously supporting the development of the positive cycle separator, Andrew Goldberg for his negative cycle detection code, and the two anonymous referees for helpful suggestions.

References

P. K. Agarwal, M. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. *Comput. Geom.*, 11:209–218, 1998.

M. Bartusch, R. H. Möhring, and F. J. Radermacher. Scheduling project networks with resource constraints and time windows. *Ann. Oper. Res.*, 16:201–240, 1988.

J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Trans. Graph.*, 14(3):203–232, 1995.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

R. G. Cromley. A spatial allocation analysis of the point annotation problem. In *Proc. 2nd Internat. Symp. on Spatial Data Handling*, pages 38–49, 1986.

M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Symp. Comput. Geom. (SoCG '91)*, pages 281–288, 1991.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, U.S.A., 1979.

M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified *NP*-complete graph problems. *Theor. Comput. Sci.*, 1:237–267, 1976.

R. Hassin. Approximation schemes for the restricted shortest path problem. *Math. Oper. Res.*, 1992.

CPLEX. *CPLEX 6.5 Reference Manual*. ILOG, 1999.

C. Iturriaga. *Map Labeling Problems*. PhD thesis, Univ. of Waterloo, Canada, 1999.

C. Iturriaga and A. Lubiw. *NP*-hardness of some map labeling problems. Technical Report CS-97-18, Univ. of Waterloo, Canada, 1997.

T. Kato and H. Imai. The *NP*-completeness of the character placement problem of 2 or 3 degrees of freedom. In *Record of Joint Conf. of Electrical and Electronic Engineers*, page 1138, Kyushu, Japan, 1988.

G. W. Klau. *A Combinatorial Approach to Orthogonal Placement Problems*. PhD thesis, Univ. d. Saarlandes, Saarbrücken, Germany, September 2001.

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 Rep., 1999a. Springer.

G. W. Klau and P. Mutzel. Optimal compaction of orthogonal grid drawings. In G. P. Cornuéjols, R. E. Burkard, and G. J. Woeginger, editors, *Proc. 7th Int. Integer Programming and Combinatorial Optimization Conf. (IPCO '99)*, volume 1610 of *LNCS*, pages 304–319, Graz, Austria, 1999b. Springer.

L. Kučera, K. Mehlhorn, B. Preis, and E. Schwarzenecker. Exact algorithms for a geometric packing problem. In *Proc. 10th Sympos. Theoret. Aspects Comput. Sci. (STACS '93)*, volume 665 of *LNCS*, pages 317–322. Springer, 1993.

J. Marks and S. Shieber. The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard Univ., Cambridge, MA, U.S.A., 1991.

K. Mehlhorn and S. Näher. *LEDA. A Platform for Combinatorial and Geometric Computing*. Cambridge Univ. Press, Cambridge, UK, 1999.

C. A. Philips. The network inhibition problem. In *Proc. 25th Ann. Symp. on Theory of Comp.*, pages 776–785. ACM, 1993.

T. Strijk and M. van Kreveld. Practical extensions of point labeling in the slider model. In *Proc. 7th ACM Symp. Adv. Geogr. Inform. Syst.*, pages 47–52, 1999.

M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Comput. Geom.*, 13:21–47, 1999.

B. Verweij and K. Aardal. An optimisation algorithm for maximum independent set with applications in map labelling. In *Proc. 7th Annu. Europ. Symp. Algorithms (ESA '99)*, volume 1643 of *LNCS*, pages 426–437, Prague, Czech Rep., 1999. Springer.

G. J. Woeginger, 2000. Inst. f. Mathematik, TU Graz, Austria. Personal communication.

A. Wolff and T. Strijk. The map labeling bibliography. URL `http://www.math-inf.uni-greifswald.de/map-labeling/bibliography`.

S. Zoraster. The solution of large 0-1 integer programming problems encountered in automated cartography. *Oper. Res.*, 38(5):752–759, 1990.