BEYOND MAXIMUM INDEPENDENT SET: AN EXTENDED MODEL FOR POINT-FEATURE LABEL PLACEMENT

Jan-Henrik Haunert[a*] Alexander Wolff[b]

[a] Geoinformatics Group, Institut für Informatik, Universität Osnabrück, Germany – janhhaunert@uni-osnabrueck.de
[b] Lehrstuhl für Informatik I, Universität Würzburg, Germany

Commission II, WG II/2

KEY WORDS: Map labeling, point-feature label placement, NP-hard, integer linear programming, cartographic requirements

ABSTRACT:

Map labeling is a classical problem of cartography that has frequently been approached by combinatorial optimization. Given a set of features in the map and for each feature a set of label candidates, a common problem is to select an independent set of labels (that is, a labeling without label–label overlaps) that contains as many labels as possible and at most one label for each feature. To obtain solutions of high cartographic quality, the labels can be weighted and one can maximize the total weight (rather than the number) of the selected labels. We argue, however, that when maximizing the weight of the labeling, interdependences between labels are insufficiently addressed. Furthermore, in a maximum-weight labeling, the labels tend to be densely packed and thus the map background can be occluded too much. We propose extensions of an existing model to overcome these limitations. Since even without our extensions the problem is NP-hard, we cannot hope for an efficient exact algorithm for the problem. Therefore, we present a formalization of our model as an integer linear program (ILP). This allows us to compute optimal solutions in reasonable time, which we demonstrate for randomly generated instances.

1. INTRODUCTION

Maps and technical drawings are usually annotated with textual or pictorial labels in order to help their users understand what they see. A natural requirement to ensure readability is that labels must not overlap. Especially in printed maps where extra information cannot be added on demand, one usually wants to label as many *features* (such as cities, rivers, or countries) as possible. Let's formalize this problem; the *basic map-labeling problem*. Given a set of features and, for each feature, a set of label candidates, each with a positive weight, select for each feature at most one label from the set of candidates such that (i) no two selected labels overlap and (ii) the sum of the weights of the selected labels is maximized. If a labeling satisfies (i), it is an independent set in the label–label conflict graph that contains a node for each label and an edge for each pair of overlapping labels. For interactive applications such as car navigation, this problem is of great importance. For high-quality cartographic purposes, however, we will argue that the above basic problem lacks expressiveness.

First attempts to formalize the rules that govern the labeling of a map have been made in the early 1960s, by Imhof (1975) and Alinhac (1962). First, cartographers (such as Yoeli 1972), but later also computer scientists tried to automate the tedious process of labeling maps using heuristics, rule-based or expert systems (Doerschler and Freeman 1989). Theoreticians focused on apparently simple special cases such as the point-feature label placement problem with four axis-aligned rectangular label candidates per point, the so-called *four-position model.* According to this model, each label must be placed such that one of its four corners coincides with the point to be labeled. Formann and Wagner (1991) showed that the problem is NP-hard and gave approximation algorithms, which guarantee that a solution "weighs" at least a certain fraction of an (unknown) optimum solution. Around the same time, researchers such as Zoraster (1986, 1990) started using integer linear programming for map labeling. Many combinatorial optimization problems can be expressed as integer linear programs (ILPs). Some of these problems are NP-hard, and hence, so is general integer linear programming. Still, highly optimized solvers for ILPs are available, and allow the user to solve small- and medium-size instances of many problems quite fast.

Zoraster's early ILP formulations were later refined by Verweij and Aardal (1999). In order to speed up the computation of optimal or near-optimal solutions, they suggested to employ so-called *cutting plane techniques*. Ribeiro and Lorena (2008) presented two ILP formulations and reported that, for instances of 1000 points, "no optimal solution was found in several hours until CPLEX [7.5] reached an out-of-memory state". Therefore, they suggested to use a so-called Lagrangian relaxation based on a clustering of the label–label conflict graph. Combining this Lagrangian clustering with the better of their two ILP formulations helped them compute, within about one hour, solutions that were no more than 1.3 % from optimal.

Rylov and Reimer (2014) formulate the basic map labeling problem as an ILP. Additionally, they express the cartographic quality of a labeling and adjust the label weights accordingly. They solve real-world instances using three different general-purpose optimization heuristics; namely greedy, gradient descent, and simulated annealing. The latter is the slowest method but yields also the best results in terms of quality.

Our contribution. We extend the ILP of Rylov and Reimer (2014) in order to take into account new cartographic aspects of a labeling such as the inter-label distance or the density of a labeling. Our model allows all criteria of the model of Rylov and Reimer (2014) to be expressed, yet in our experiments we used a simpler weight function in which the weight of a label depends only on the corresponding feature and not on the position of the label. We also do not rule out labels that intersect coast lines or other significant features of the map background; such label candidates can easily be detected in a pre-processing step. Commercial solvers (such as CPLEX or Gurobi) can solve instances

*Corresponding author

of our extended ILP of more than 1000 labels to optimality, even if the label–label conflict graph is very dense.

2. METHODOLOGY

We start by recalling (see Sect. 2.1) and discussing (see Sect. 2.2) the ILP of Rylov and Reimer (2014), which is the basis of our extensions (see Sects. 2.3–2.5. We discuss how to set up and solve our ILP (see Sect. 2.6).

2.1 A Basic Integer Linear Program

In the basic map-labeling problem, we are given a set $\mathcal{F}$ of features and, for each feature $f \in \mathcal{F}$, a set $L(f)$ of label candidates. The set of all label candidates is $L = \bigcup_{f\in\mathcal{F}} L(f)$. Rylov and Reimer (2014) introduce a 0–1 variable for each label candidate:

$$x_\ell \in \{0,1\} \quad \text{for each } \ell \in L\,. \tag{1}$$

Since we will later add another group of variables, we refer to the above variables as *x-variables*. Generally, if a variable x_ℓ is set to 1, the label candidate ℓ is *selected*; it becomes a *label*. (For simplicity, in the sequel, by "labels" we will also mean label candidates.) Using these variables, Rylov and Reimer define the following objective function, which aims at selecting a subset of the label candidates of maximum total weight.

$$\text{Maximize} \sum_{\ell\in L} w(\ell)\cdot x_\ell \tag{2}$$

They forbid that two overlapping labels are selected by requiring

$$x_a + x_b \le 1 \quad \text{for each } a, b \in L \text{ with } a \ne b \text{ and } a \text{ overlaps } b. \tag{3}$$

If the given labeling instance is such that any two label candidates of the same feature overlap, then Constraint (3) automatically ensures that any feature receives at most one label. If, however, a feature has label candidates that do not overlap each other, the following constraint needs to be added:

$$\sum_{\ell\in L(f)} x_\ell \le 1 \quad \text{for each } f \in \mathcal{F} \tag{4}$$

Rylov and Reimer term their ILP formulation a "comprehensive multi-criteria model". Indeed, the model is quite general since many criteria of cartographic quality can be modeled by specifying the weight function w. We argue, however, that not all criteria of cartographic quality can be expressed with this. Nevertheless, the ILP presented in this section forms the basis for our extended model, which is why we term it the *basic ILP*. It has $O(|L|)$ integer variables and $O(|L|^2)$ constraints since we may need to instantiate Constraint (3) for each pair of label candidates.

2.2 Disadvantages and advantages of the basic ILP

A shortcoming of the basic ILP is that the weight of a label needs to be defined before solving the ILP and, thus, has to be set without knowing which of the other labels are placed. Therefore, the possibility of expressing interdependences between labels is very limited. For example, to judge whether the association between a point label ℓ and its corresponding point p is clear, it is important to know which other points are labeled. This particularly holds if the rule is to display only those points that receive a label: With that rule, only if a point p' is labeled, a label ℓ can be misinterpreted as the label for p'. We could rule out ambiguous label–point associations by setting up hard constraints similar to Constraint (3), for example, to express that a label ℓ must never be placed together with a label for p'. We argue, however, that such a constraint would be too strict and that the clarity of associations should rather be modeled in the objective function. This is not possible with a weighted sum of the x-variables.

A second problem with the basic ILP is that, among two feasible solutions $S_1 \subseteq L$ and $S_2 \subseteq L$ with $S_1 \subsetneq S_2$, the solution S_2 containing more labels is always preferred. In other words, if a label can be added to a solution, then this solution cannot be optimal. Consequently, an optimal labeling tends to be densely packed with labels and may occlude the map background almost completely. In fact, in the optimization community, map labeling is referred to as a *packing problem* (Formann and Wagner 1991). It is clearly not the objective of map labeling, however, to occlude the map background as much as possible. Therefore, the existing models need to be reconsidered.

Before we turn to an extension of the model, we need to admit that the basic ILP is not as bad as it may seem.

First of all, every subset of an optimal solution to the basic ILP is a feasible solution. Therefore, it can be reasonable to first compute an optimal (or a close-to-optimal) solution $S \subseteq L$ to the basic ILP, to store this solution, and, at any time, to display only a selection S' of rectangles from S, such that the map background is not occluded too much. This two-step approach of first computing a densely packed solution S without label–label overlaps and then filtering S to obtain the actual labeling S' is particularly useful for real-time cartography. In this application, labelings need to be computed on the fly and thus the supposedly difficult part of the labeling problem – avoiding label–label overlaps without loosing too much information – arguably should be solved in a pre-processing step (Been et al. 2006).

A second argument for the basic ILP is that the occlusion of the map background can be controlled by defining the labels larger than necessary. For example, the label holding a certain text can be defined by buffering the text's bounding box with a certain distance $\varepsilon \in \mathbb{R}_{>0}$. This in essence means that between any two displayed texts a minimum distance of 2ε is ensured. We argue that strictly requiring a minimum distance between two texts is important to ensure the legibility of each label. To avoid that large parts of the map are covered with labels, however, a more global constraint is needed.

2.3 Penalizing ambiguous label–feature associations

In this section we present an extension of the basic ILP to reduce the risk that a user interprets a label of some feature f as the label of a different feature f'. We first discuss this in a general context and then exemplify it for point-feature label placement.

Our model is based on a graph $G = (L, E)$ that contains a node for each label in L. The set E contains an edge between two labels a, b if placing both a and b is possible yet undesirable. To express how undesirable it is to place both a and b, we define a cost function $c\colon E \to \mathbb{R}_{>0}$. If for an edge $e = \{a, b\} \in E$ both labels a, b are selected at the same time, we charge the cost $c(e)$, meaning that the objective value is reduced by $c(e)$. For ease of notation, we write $c(a, b)$ for $c(e)$ with $e = \{a, b\}$. To formalize the idea in linear terms, we introduce continuous variables

$$y_e \in [0,1] \quad \text{for each } e \in E\,, \tag{5}$$

which we term *y-variables*. These are linked with the x-variables such that $y_e = 1$ if both $x_a = 1$ and $x_b = 1$ for $e = \{a, b\}$:

$$y_e \ge x_a + x_b - 1 \quad \text{for each } e = \{a, b\} \in E\,. \tag{6}$$

If at least one of the two labels a, b is not selected, however, Constraint (6) does not have any influence on the variable y_e.

The extended model has the following objective function:

$$\text{Maximize} \sum_{\ell \in L} w(\ell) \cdot x_\ell - \sum_{e \in E} c(e) \cdot y_e \tag{7}$$

Since $x_a = 1$ and $x_b = 1$ entails $y_e = 1$ for $e = \{a, b\}$, the cost $c(e)$ is correctly charged if a and b are selected. Else, if $x_a = 0$ or $x_b = 0$, y_e is free within the interval $[0, 1]$; since y_e appears with a negative coefficient in an objective function that we maximize, it will receive the smallest possible value, which is 0, and the cost $c(e)$ for e will not be charged. This means that, even though we defined y_e as a continuous variable, it will always receive the value 0 or 1 in an optimal solution. We conclude that our extended model does not need additional integer variables, which has computational advantages. Since our model has both integer variables and continuous variables it is in fact a *mixed* integer linear program (MILP).

Finally, we suggest a cost function c when labeling points in the four-position model. In this case, we would like to prevent situations such as the one displayed in Fig. 1. In this example, the rectangular label ℓ corresponds to the point p, yet a user may think that ℓ is the label for q, since q is within a specified distance $\rho \in \mathbb{R}_{>0}$ from ℓ. Therefore, ℓ should not be displayed together with q. Assuming that q is displayed if and only if it receives a label, we can equally well say that ℓ should not be displayed together with one of the labels for q. Since the upper-right label for q is not possible in conjunction with ℓ, we add three edges to the graph G, namely $\{\ell, r\}$, $\{\ell, s\}$, and $\{\ell, t\}$.

For edges $\{\ell, r\}$ and $\{\ell, s\}$, we define the costs $c(\ell, r) = c(\ell, s) = \alpha \cdot w(\ell)$, where $\alpha \in [0, 1)$ is a user-selectable parameter. For example, $\alpha = 0.4$ means that selecting ℓ with r or s is 40% less profitable than selecting ℓ without r or s. For edge $\{\ell, t\}$, we define the cost $c(\ell, t) = \alpha \cdot w(\ell) + \alpha \cdot w(t)$ since not only q is within distance ρ from ℓ but also p is within distance ρ from t.

2.4 Controlling occlusion of the map background

In Sect. 2.2 we argued that the occlusion of the map background could be controlled by defining the labels larger than necessary, for example, by buffering the bounding boxes for the texts that are to be displayed. Because of Constraint (3), this in essence means that any two labels have to be separated with a certain amount of empty space between them. While Constraint (3) acts locally, we may also require that, overall, at most a certain number $K_{\text{global}} \in \mathbb{Z}_{>0}$ of labels can be placed. This can be modeled as follows.

$$\sum_{\ell \in L} x_\ell \le K_{\text{global}} \tag{8}$$

Constraint (8) acts on the entire map and thus on a global level. To control the occlusion of the map background in an adequate way, we introduce a constraint whose region of influence can be adjusted, such that this more general constraint subsumes both Constraint (3) and Constraint (8), but also constraints acting somehow between the local and the global level. More precisely, our model relies on the definition of a region $\Gamma \subseteq \mathbb{R}^2$. Our idea is to require that every translate γ of Γ overlaps at most a certain number K_Γ of labels. For example, when defining Γ as a disk of constant radius δ, our requirement means the following: In the output map, every disk of radius δ overlaps at most K_Γ labels. By restricting $\Gamma = \{(0, 0)\}$ to the origin and setting $K_\Gamma = 1$, we can avoid overlaps between any two labels and thus achieve the same effect as with Constraint (3). On the other hand, by setting $\Gamma = \mathbb{R}^2$ and

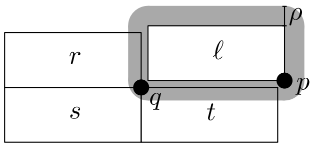


Figure 1: A point p with a label ℓ, which can be misinterpreted as the label for q, since q is within distance ρ from ℓ. When labeling p with ℓ, the label for q can be r, s, or t.

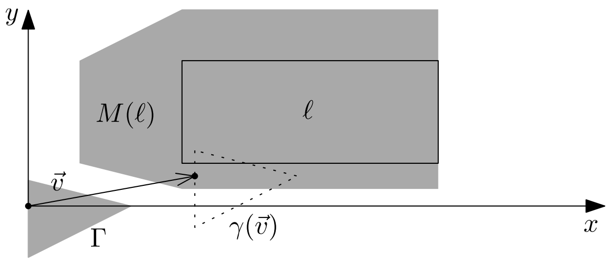


Figure 2: A label ℓ, a region Γ, and the Minkowski difference $M(\ell) = \ell \ominus \Gamma$. Translating Γ with a vector $\vec{v} \in M(\ell)$ yields a region $\gamma(\vec{v})$ overlapping ℓ.

$K_\Gamma = K_{\text{global}}$, we allow at most K_{global} labels to be placed in total, which is the same as Constraint (8).

To formalize our requirement in linear terms, a first idea is to define a constraint similar to Constraint (8) for each region γ that can be obtained by translating Γ. In other words, we would need one constraint for each possible translation vector $\vec{v} \in \mathbb{R}^2$. Let $\gamma(\vec{v}) = \Gamma + \vec{v} \subseteq \mathbb{R}^2$ be the region obtained by translating Γ with $\vec{v}$, and let $L(\gamma) \subseteq L$ be the set of labels in L overlapping a region $\gamma \subseteq \mathbb{R}^2$. Then, we can express our requirement as follows.

$$\sum_{\ell \in L(\gamma(\vec{v}))} x_\ell \le K_\Gamma \quad \text{for each } \vec{v} \in \mathbb{R}^2 \tag{9}$$

The problem with this definition is that there are infinitely many vectors in $\mathbb{R}^2$, thus it seems that we need an infinite number of instantiations of Constraint (9). This is not the case, however, since for two distinct vectors $\vec{v}_1, \vec{v}_2 \in \mathbb{R}^2$ it can hold that $L(\gamma(\vec{v}_1)) = L(\gamma(\vec{v}_2))$. In this case, Constraint (9) would clearly be the same for both $\vec{v} = \vec{v}_1$ and $\vec{v} = \vec{v}_2$, thus we need not instantiate it twice. Generally, there can only be $2^{|L|}$ distinct instantiations of Constraint (9), which is a finite (though large) number.

In the following, we assume that the labels as well as the region $\Gamma \subseteq \mathbb{R}^2$ have constant complexity, for example, they are axis-aligned rectangles (each described with four parameters) or disks (each described with three parameters). We will see that, under these assumptions, the extended model with our general requirement has the same asymptotic number of constraints as the basic ILP. In the examples that we present in this section, we will use an irregular triangle for Γ to show that Γ is not required to be symmetric. Constant complexity of shapes is required, however, to ensure that our ILP has a manageable size.

To formalize our requirement, we consider the labels and Γ as sets of position vectors. We denote the Minkowski difference (de Berg et al. 2008) of a label ℓ and Γ as $M(\ell)$, that is,

$$M(\ell) = \ell \ominus \Gamma = \{\vec{a} - \vec{b} \mid \vec{a} \in \ell, \vec{b} \in \Gamma\}\,. \tag{10}$$

Just as ℓ and Γ, $M(\ell)$ is a set of vectors that can be interpreted as a region; see Fig. 2. Moreover, $M(\ell)$ is the set of all vectors such that translating Γ with a vector $\vec{v} \in M(\ell)$ yields a region $\gamma(\vec{v})$ overlapping ℓ. Let us now consider two labels ℓ_1, ℓ_2 and the Minkowski differences $M(\ell_1) = \ell_1 \ominus \Gamma$ and $M(\ell_2) = \ell_2 \ominus \Gamma$.

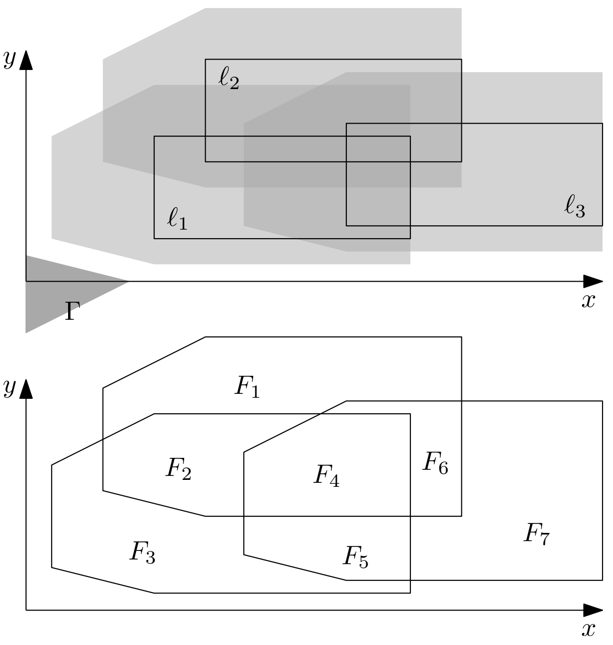


Figure 3: Three labels ℓ_1, ℓ_2, and ℓ_3 with their Minkowski differences $M(\ell_1)$, $M(\ell_2)$, and $M(\ell_3)$ (top) as well as the facets of the arrangement resulting from the overlay of $M(\ell_1)$, $M(\ell_2)$, and $M(\ell_3)$ (bottom).

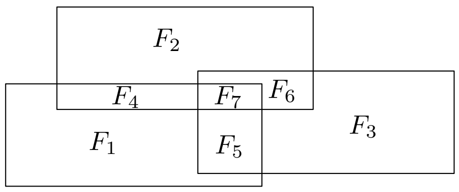


Figure 4: The seven facets of the arrangement formed by the three labels ℓ_1, ℓ_2, ℓ_3 in Fig. 3.

The set $M(\ell_1) \cap M(\ell_2)$ contains every vector $\vec{v}$ such that $\gamma(\vec{v})$ overlaps both ℓ_1 and ℓ_2. We can easily generalize this idea to any set $L' \subseteq L$ of labels: Each region $\gamma(\vec{v})$ with $\vec{v} \in \bigcap_{\ell \in L'} M(\ell)$ overlaps all labels in L'.

To find each relevant subset $L' \subseteq L$, that is, each set of labels for which we need to set up a constraint similar to Constraint (9), we consider the arrangement $\mathcal{A}$ formed by the boundaries of the region $M(\ell)$ for each label $\ell \in L$; see Fig. 3. More formally, we consider $\mathcal{A}$ as a set of facets. For a facet $F \in \mathcal{A}$ and any two vectors $\vec{v}_1, \vec{v}_2 \in F$, it holds that the regions $\gamma(\vec{v}_1)$ and $\gamma(\vec{v}_2)$ overlap the same labels, that is, $L(\gamma(\vec{v}_1)) = L(\gamma(\vec{v}_2))$. Therefore, we need only one constraint for each facet of $\mathcal{A}$:

$$\sum_{\ell \in L(\gamma(\vec{v}))} x_\ell \leq K_\Gamma \quad \text{for each facet } F \in \mathcal{A}\,, \tag{11}$$

where $\vec{v}$ is an arbitrary vector in F.

Constraint (11) has the same "meaning" as Constraint (9), yet it is defined on the discrete set $\mathcal{A}$ of facets rather than on the continuous set $\mathbb{R}^2$ of all possible translations of Γ. The number of facets of $\mathcal{A}$ equals the number of instantiations of Constraint (11). Recall that by our assumption each label $\ell \in L$ and the region Γ have constant complexity. Then the same holds for $M(\ell) = \ell \ominus \Gamma$, which yields that the total number of vertices and line segments of $\mathcal{A}$ is $O(|L|^2)$. Since $\mathcal{A}$ is planar, the number of facets is linear in the number of vertices, thus $\mathcal{A}$ has $O(|L|^2)$ facets.

We conclude that Constraint (11) requires $O(|L|^2)$ instantiations – one for each facet of $\mathcal{A}$ – and thus (asymptotically) the extended model does not require more constraints than the basic ILP. Furthermore, our model requires the same $|L|$ integer variables (x_ℓ for each $\ell \in L$) as the basic ILP. On the other hand, we need $O(|L|^2)$ additional continuous variables (y_e for each $e \in E$), since we may need to charge a cost for each pair of labels. In practice, however, it is reasonable to assume that the graph G is sparse, since only a few labels (probably constant number of labels) will interfere with a label. Therefore, it is reasonable to assume that the number of edges of G is in $O(|L|)$ and thus our ILP has the same size as the basic ILP.

2.5 Computational advantages of the extended model

Most ILP solvers rely on a strategy termed *branch and bound*. Such solvers usually start solving a model by solving the model's LP relaxation. This generally means that the integer variables of the model are allowed to receive fractional values – in our example, $x_\ell \in \{0,1\}$ is relaxed to $x_\ell \in [0,1]$ for each $\ell \in L$. In terms of the objective function, an optimal solution to the ILP can not be better than an optimal solution to the corresponding LP relaxation. Therefore, if the aim is to maximize the objective function, the LP relaxation offers an upper bound for the objective function of the ILP. The success of the existing solvers usually depends on how tight this upper bound is, that is, on how close it is to the objective value of an optimal integer solution. The tighter the upper bound is, the more likely it is that the solver can prune branches of the search tree (the so-called *branch-and-bound tree*) that is explored to find an optimal integer solution. Computing bounds by solving the model's LP relaxation is reasonable since LPs (other than ILPs) can be solved efficiently both in theory (for example, with interior point methods) and in practice (usually with the simplex algorithm).

A problem with the basic ILP is that its LP relaxation does not offer a good upper bound. Suppose, for example, that we need to solve the instance in Fig. 3 with the three rectangular labels ℓ_1, ℓ_2, and ℓ_3, each of which belongs to a different feature. To rule out overlaps between the labels, we need to instantiate Constraint (3) three times: $x_{\ell_1} + x_{\ell_2} \leq 1$, $x_{\ell_1} + x_{\ell_3} \leq 1$, and $x_{\ell_2} + x_{\ell_3} \leq 1$. Assuming unit weights (that is, $w(\ell) = 1$ for any $\ell \in L$), an optimal integer solution $S^\star$ is obtained by setting an arbitrary variable to 1. Such a solution has objective value 1. In the LP relaxation, however, we can satisfy Constraint (3) by setting each of the three variables to 0.5. This solution S' has objective value 1.5, which is a rather poor estimate for the objective value of $S^\star$.

As mentioned in Sect. 2.4, we can alternatively avoid overlapping labels by setting up Constraint (11) with a particular choice of Γ and K_Γ. More precisely, we restrict $\Gamma = \{(0,0)\}$ to the origin of the coordinate system. This yields $M(\ell) = \ell$ for every label $\ell \in L$ and the arrangement $\mathcal{A}$ simply results from an overlay of all labels. By setting $K_\Gamma = 1$, we achieve that every facet of $\mathcal{A}$ is covered by at most one label. Hence, no two labels overlap. Using the same example as before, we obtain an arrangement of $|\mathcal{A}| = 7$ facets (not counting the unbounded exterior face); see Fig. 4. This results in the following seven instantiations of Constraint (11): $F_1\colon x_{\ell_1} \leq 1$; $F_2\colon x_{\ell_2} \leq 1$; $F_3\colon x_{\ell_3} \leq 1$; $F_4\colon x_{\ell_1} + x_{\ell_2} \leq 1$; $F_5\colon x_{\ell_1} + x_{\ell_3} \leq 1$; $F_6\colon x_{\ell_2} + x_{\ell_3} \leq 1$; $F_7\colon x_{\ell_1} + x_{\ell_2} + x_{\ell_3} \leq 1$. Note that the inequality for F_7 implies all other inequalities, thus we can avoid overlaps between labels with a single constraint. In the LP relaxation, this constraint clearly forbids setting all variables to 0.5. It turns out that, in this example, an optimal solution of the LP relaxation (for example, the solution $x_{\ell_1} = x_{\ell_2} = x_{\ell_3} = 1/3$) has the same value (that is, 1) as an optimal solution of the ILP.

Generally, the LP relaxation of the extended model yields an upper bound that is at least as tight as the upper bound yielded by the LP relaxation of the basic ILP. Therefore, when ruling out overlaps between labels with Constraint (11), we can expect better running times than with Constraint (3).

2.6 Solving the ILPs

Exact algorithms for solving ILPs such as branch and bound and an extension termed *branch and cut* are readily applicable with commercial ILP solvers, for example, CPLEX or Gurobi. As all exact algorithms for NP-hard problems, those algorithms have an exponential worst-case running time. Nevertheless, we choose this approach, not least to generate benchmark solutions for the evaluation of faster heuristics, which we plan to develop with future research. Though using such solvers is relatively straight forward, some algorithmic engineering is necessary to set up the constraints. In particular, for Constraint (11), we need to compute the arrangement $\mathcal{A}$, which can be done efficiently with a plane-sweep algorithm (de Berg et al. 2008).

3. EXPERIMENTAL RESULTS

We implemented our algorithms in Java and tested them with respect to their running times in practice. We used the solver Gurobi, version 6.0, to solve our ILPs. To compute the arrangement $\mathcal{A}$ for Constraint (11) and to find overlapping labels, we used the programming library JTS Topology Suite. All tests were performed on a Windows PC with an Intel Core i5-3570 CPU and 8 GB of RAM.

We constructed the instances for our tests based on point sets of different sizes, which we generated randomly. For each point, we generated four rectangular labels, which we placed according to the four position model. We gradually increased the number $|L|$ of labels without changing their spatial density, which means that we increased the available area linearly with $|L|$.

More precisely, we tested our exact method for randomly generated instances with $|L| = 400, 800, \dots, 2400$ rectangular labels. For each instance, we sampled $|L|/4$ points uniformly at random in the square $[0, \sqrt{|L|/4}] \times [0, \sqrt{|L|/4}]$, which implies on average one point per unit area, and we generated four rectangular labels of width 1 and height 0.5 for each point, using the four position model. Afterwards, we slightly enlarged the rectangles, by offsetting their sides 0.01 units outwards. Due to this enlargement, any two labels for the same point overlap. As a consequence, Constraint (4) is automatically satisfied in every solution without overlapping labels. Therefore, we did not instantiate that constraint in our tests. Figure 5 shows three instances that we generated this way.

To define the quality of a solution, we sampled the weight $w(\ell)$ for each label $\ell \in L$ uniformly at random from the interval $[0, 1]$. Furthermore, we defined $\rho = 0.02$ and $\alpha = 0.5$. (Recall that we charge a cost of $\alpha \cdot w(\ell)$ for any labeled point that is within distance ρ from a label of a different point.) Finally, to test the effect of Constraint (11), we defined that every axis-aligned square of size 5×5 must not overlap more than $K_\Gamma = 10$ selected labels. Table 1 summarizes the results of our methods.

The first method, which we term ILP1, is based on an ILP using

- Constraint (3) to avoid overlapping labels,
- Constraint (6) to link the x-variables with the y-variables,
- Constraint (11) to ensure that every axis-aligned square of size 5×5 overlaps at most $K_\Gamma = 10$ labels, and
- Objective (7), which considers both weights of labels and costs for ambiguous label–point associations.

Table 1: Experimental results for instances with different numbers but constant spatial density of rectangles: Number $|L|$ of rectangles; average running times in seconds (for setting up and solving the ILP, and in total) with different ILP formulations. For each row, ten instances were tested.

	ILP1			ILP2		
$\|L\|$	set up	solution	total	set up	solution	total
400	1.3	0.7	2.0	1.4	0.4	1.7
800	4.6	6.9	11.5	5.1	4.5	9.6
1200	10.7	15.2	25.9	11.7	11.8	23.5
1600	24.9	115.3	140.3	26.3	98.4	124.7
2000	32.2	290.4	322.7	36.3	314.9	351.3
2400	61.1	3158.3	3219.5	67.0	2801.5	2868.5

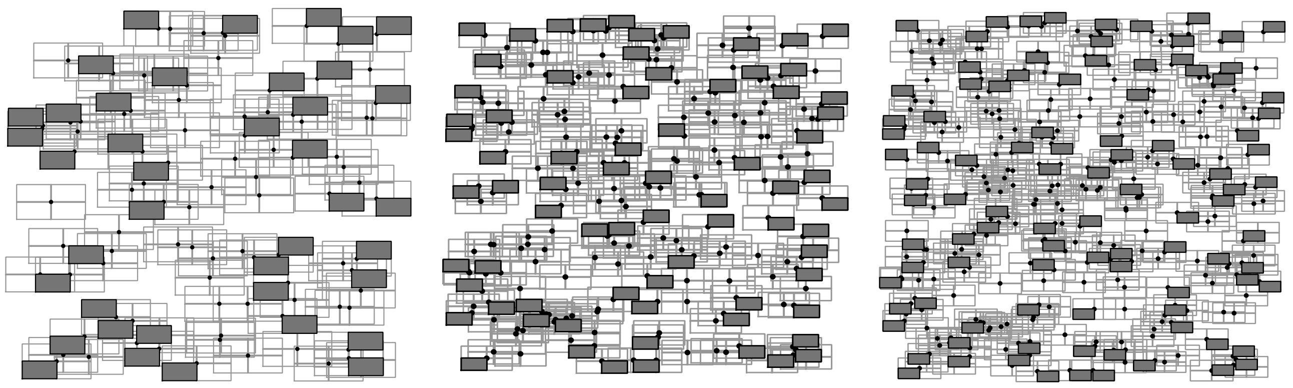
For setting up ILP1 we need to find overlapping labels for Constraint (3) and to compute the arrangement for Constraint (11). This requires a considerable amount of time, which is shown in the second column of Table 1. For larger instances, however, the time for the set up of the ILP is clearly dominated by the time for the solution of the ILP, which is shown in the third column of Table 1. Solving an instance with 2400 labels requires almost one hour, which is certainly unacceptable in practice.

The second method, which we term ILP2, is very similar to ILP1. The only difference is that ILP2 does not use Constraint (6) to avoid overlapping labels. Instead, it uses Constraint (11) with $\Gamma = \{(0, 0)\}$ and $K_\Gamma = 1$. Recall that this means that every point in the plane is contained in at most one selected label, which is the same as forbidding label–label overlaps. As ILP1, ILP2 also uses Constraint (11) to avoid more than 10 labels in any square of size 5×5, thus ILP2 uses Constraint (11) with two different settings for Γ and K_Γ, and it requires two arrangements to be computed. Therefore, the set-up time for ILP2 is generally higher than for ILP1. On the other hand, the solution time for ILP2 is usually lower, which can be explained with the fact that its LP relaxation yields a tighter upper bound for the objective value of an optimal ILP solution; see Sect. 2.5. Considering the total running times with ILP1 and ILP2, our experiments do not allow for a clear conclusion in favor of one or the other method. ILP2 seems to offer a small advantage, though.

4. CONCLUSION

We conclude that with our extended model we can effectively control the density of the labels in the output map and thus avoid situations in which the background map is occluded too much. Furthermore, our model improves the clarity of label–point associations. If the input point set is not too dense, our ILP-based approach yields optimal solutions in reasonable time. With respect to cartographic quality, it would be interesting to see how our model extension performs with a more sophisticated weight function such as (Rylov and Reimer 2014).

To process large instances fast, we consider it promising to apply an LP-based rounding heuristic. Such a heuristic relies on a strong LP-relaxation, which the ILP formulation with constraints for pairs of conflicting labels (i.e., ILP1) does not offer. Our arrangement-based ILP formulation (i.e., ILP2) has this favorable property, however, and it does not require substantially more constraints. More consideration on this aspect will we given in a journal article, which we plan to publish in the near future. First results that we obtained with ILP2 and an LP-based heuristic rounding are promising.



(a) 100 points in a square of size 10×10 with $|L| = 400$ labels

(b) 200 points in a square of size 14.1×14.1 with $|L| = 800$ labels

(c) 300 points in a square of size 17.3×17.3 with $|L| = 1200$ labels

Figure 5: Three instances with on average one point per unit area. Labels selected in an optimal solution are filled gray. Every axis-aligned square of size 5×5 overlaps at most $K_\Gamma = 10$ selected labels.

References

1 Alinhac, G., 1962. Cartographie Théorique et Technique. Institut Géographique National, Paris, chapter IV.

2 Been, K., Daiches, E. and Yap, C., 2006. Dynamic map labeling. IEEE Trans. Visual. Comput. Graphics 12(5), pp. 773–780.

3 de Berg, M., Cheong, O., van Kreveld, M. and Overmars, M., 2008. Computational Geometry: Algorithms and Applications. 3rd edn, Springer TELOS, Santa Clara, CA, USA.

4 Doerschler, J. S. and Freeman, H., 1989. An expert system for dense-map name placement. In: Proc. Auto-Carto 9, pp. 215–224.

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

6 Imhof, E., 1975. Positioning names on maps. The American Cartographer 2(2), pp. 128–144.

7 Ribeiro, G. M. and Lorena, L. A. N., 2008. Lagrangean relaxation with clusters for point-feature cartographic label placement problems. Comput. Oper. Res. 35(7), pp. 2129–2140.

8 Rylov, M. A. and Reimer, A. W., 2014. A comprehensive multi-criteria model for high cartographic quality point-feature label placement. Cartographica 49(1), pp. 52–68.

9 Verweij, B. and Aardal, K., 1999. An optimisation algorithm for maximum independent set with applications in map labelling. In: Proc. 7th Annu. European Symp. Algorithms (ESA'99), LNCS, Vol. 1643, Springer-Verlag, pp. 426–437.

10 Yoeli, P., 1972. The logic of automated map lettering. The Cartographic Journal 9, pp. 99–108.

11 Zoraster, S., 1986. Integer programming applied to the map label placement problem. Cartographica 23(3), pp. 16–27.

12 Zoraster, S., 1990. The solution of large 0-1 integer programming problems encountered in automated cartography. Operations Research 38(5), pp. 752–759.