Label Placement in Road Maps

Andreas Gemsa, Benjamin Niedermann, and Martin Nöllenburg

Institute of Theoretical Informatics, Karlsruhe Institute of Technology, Germany

Abstract. A road map can be interpreted as a graph embedded in the plane, in which each vertex corresponds to a road junction and each edge to a particular road section. We consider the cartographic problem to place non-overlapping road labels along the edges so that as many road sections as possible are identified by their name, i.e., covered by a label. We show that this is NP-hard in general, but the problem can be solved in polynomial time if the road map is an embedded tree.

1 Introduction

Map labeling is a well-known cartographic problem in computational geometry [12, Chapter 58.3.1], [14]. Depending on the type of map features, one can distinguish labeling of *points*, *lines*, and *areas*. Common cartographic quality criteria are that labels must be disjoint and clearly identify their respective map features [8]. Most of the previous work concerns point labeling, while labeling line and area features received considerably less attention. In this paper we address labeling linear features, namely roads in a road map.

Geometrically, a *road map* is the representation of a *road graph* G as an arrangement of fat curves in the plane $\mathbb{R}^2$. Each *road* is a connected subgraph of G (typically a simple path) and each edge belongs to exactly one road. Roads may intersect each other in *junctions*, the vertices of G, and we denote an edge connecting two junctions as a *road section*. In road labeling the task is to place the road names inside the fat curves so that the road sections are identified unambiguously, see Fig. 1.

Chirié [1] presented a set of rules and quality criteria for label placement in road maps based on interviews with cartographers. This includes that (C1) labels are placed inside and parallel to the road shapes, (C2) every road section between two junctions should be clearly identified, and (C3) no two road labels may intersect. Further, he gave a mathematical description for labeling a single road and introduced a heuristic for sequentially labeling all roads in the map. Imhof's foundational cartographic work on label positioning in maps lists very similar quality criteria [4]. Edmondson et al. [2] took an algorithmic perspective on labeling a single linear feature (such as a river). While Edmondson et al. considered *non-bent* labels, Wolff et al. [13] introduced an algorithm for single linear feature that places labels following the curvature of the linear feature. Strijk [10] considered static road labeling with embedded labels and presented a heuristic for selecting non-overlapping labels out of a set of label candidates. Seibert and Unger [9] considered grid-shaped road networks. They showed that in those networks it is NP-complete and APX-hard to decide whether for every road at least one label can be placed. Yet, Neyer and Wagner [7] introduced a practically efficient algorithm that

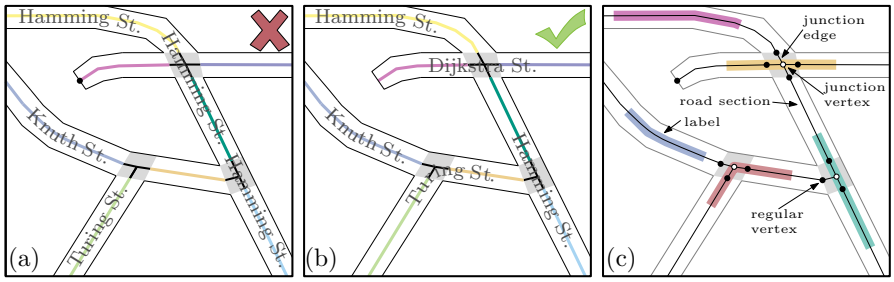

Fig. 1. a–b): Two ways to label the same road network. Each road section has its own color. Junctions are marked gray. Fig. b) identifies all road sections. c) Illustration of the road graph and relevant terms.

finds such a grid labeling if possible. Maass and Döllner [6] presented a heuristic for labeling the roads of an interactive 3D map with objects (such as buildings). Apart from label-label overlaps, they also resolve label-object occlusions. Vaaraniem et al. [11] used a force-based labeling algorithm for 2D and 3D scenes including road label placement.

Contribution. While in grid-shaped road networks it is sufficient to place a single label per road to clearly identify all its road sections, this is not the case in general road networks. Consider the example in Fig. 1. In Fig. 1a), it is not obvious whether the orange road section in the center belongs to *Knuth St.* or to *Turing St*. Simply maximizing the number of placed labels, as often done for labeling point features, can cause undesired effects like unnamed roads or clumsy label placements (e.g., around *Dijkstra St.* and *Hamming St.* in Fig. 1a)). Therefore, in contrast to Seibert and Unger [9], we aim for maximizing the number of *identified* road sections, i.e., road sections that can be clearly assigned to labels; see Fig. 1b).

Based on criteria (C1)–(C3) we introduce a new and versatile model for road labeling in Section 2. In Section 3 we show that the problem of maximizing the number of identified road sections is NP-hard for general road graphs, even if each road is a path. For the special case that the road graph is a tree, we present a polynomial-time algorithm in Section 4. This special case is not only of theoretical interest, but our algorithm in fact provides a very useful subroutine in exact or heuristic algorithms for labeling general road graphs. Our initial experiments, sketched in the full version [3], show that real-world road networks decompose into small subgraphs, a large fraction of which (more than 85.1%) are actually trees, and thus can be labeled optimally by our algorithm.

2 Preliminaries

As argued above, a road map is a collection of fat curves in the plane, each representing a particular piece of a named road. If two (or more) such curves intersect, they form junctions. A *road label* is again a fat curve (the bounding shape of the road name) that is contained in and parallel to the fat curve representing its road. We observe that labels of different roads can intersect only within junctions and that the actual width of the curves

is irrelevant, except for defining the shape and size of the junctions. These observations allow us to define the following more abstract but equivalent road map model.

A *road map* $\mathcal{M}$ is a planar *road graph* $G = (V, E)$ together with a planar embedding $\mathsf{E}(G)$, which can be thought of as the geometric representation of the road axes as thin curves; see Fig 1c). We denote the number of vertices of G by n, and the number of edges by m. Observe that since G is planar $m = O(n)$. Each edge $e \in E$ is either a *road section*, which is not part of a junction, or a *junction edge*, which is part of a junction. Each vertex $v \in V$ is either a *junction vertex* incident only to junction edges, or a *regular vertex* incident to one road section and at most one junction edge, which implies that each regular vertex has degree at most two. A junction vertex v and its incident edges are denoted as a *junction*. The edge set E decomposes into a set $\mathcal{R}$ of edge-disjoint *roads*, where each road $R \in \mathcal{R}$ induces a connected subgraph of G. Without loss of generality we assume no two road sections G are incident to the same vertex. Thus, a road decomposes into road sections, separated by junction vertices and their incident junction edges. In realistic road networks the number of roads connected passing through a junction is small and does not depend on the size of the road network. We therefore assume that each vertex in G has constant degree. We assume that each road $R \in \mathcal{R}$ has a name whose length we denote by $\lambda(R)$.

For simplicity, we identify the embedding $\mathsf{E}(G)$ with the points in the plane covered by $\mathsf{E}(G)$, i.e. $\mathsf{E}(G) \subseteq \mathbb{R}^2$. We also use $\mathsf{E}(v)$, $\mathsf{E}(e)$, and $\mathsf{E}(R)$ to denote the embeddings of a vertex v, an edge e, and a road R.

We model a label as a simple open curve $\ell\colon [0,1] \to \mathsf{E}(G)$ in $\mathsf{E}(G)$. Unless mentioned otherwise, we consider a curve ℓ always to be simple and open, i.e., ℓ has no self-intersections and its end points do not coincide. In order to ease the description, we identify a curve ℓ in $\mathsf{E}(G)$ with its image, i.e., ℓ denotes the set $\{\ell(t) \in \mathsf{E}(G) \mid t \in [0,1]\}$. The start point of ℓ is denoted as the *head* $h(\ell)$ and the endpoint as the *tail* $t(\ell)$. The length of ℓ is denoted by $\mathrm{len}(\ell)$. The curve ℓ *identifies* a road section r if $\ell \cap \mathsf{E}(r) \neq \emptyset$. For a set $\mathcal{L}$ of curves $\omega(\mathcal{L})$ is the number of road sections that are identified by the curves in $\mathcal{L}$. For a single curve ℓ we use $\omega(\ell)$ instead of $\omega(\{\ell\})$. For two curves ℓ_1 and ℓ_2 it is not necessarily true that $\omega(\{\ell_1, \ell_2\}) = \omega(\ell_1) + \omega(\ell_2)$, because they may identify the same road section twice.

A *label* ℓ for a road R is a curve $\ell \subseteq \mathsf{E}(R)$ of length $\lambda(R)$ whose endpoints must lie on road sections and not on junction edges or junction vertices. Requiring that labels end on road sections avoids ambiguous placement of labels in junctions where it is unclear how the road passes through it. A *labeling* $\mathcal{L}$ for a road map with road set $\mathcal{R}$ is a set of mutually non-overlapping labels, where we say that two labels ℓ and ℓ' *overlap* if they intersect in a point that is not their respective head or tail.

Following the cartographic quality criteria (C1)–(C3), our goal is to find a labeling $\mathcal{L}$ that maximizes the number of identified road sections, i.e., for any labeling $\mathcal{L}'$ we have $\omega(\mathcal{L}') \leq \omega(\mathcal{L})$. We call this problem MAXIDENTIFIEDROADS.

Note that assuming the road graph G to be planar is not a restriction in practice. Consider for example a road section r that overpasses another road section r', i.e., r is a bridge over r', or r' is a tunnel underneath r. In order to avoid overlaps between labels placed on r and r', we either can model the intersection of r and r' as a regular crossing of two roads or we split r' in smaller road sections that do not cross r. In both cases

the corresponding road graph becomes planar. In the latter case we may obtain more independent roads created by chopping r' into smaller pieces.

3 Computational Complexity

We first study the computational complexity of road labeling and prove NP-hardness of MAXIDENTIFIEDROADS in the following sense.

Theorem 1. *For a given road map $\mathcal{M}$ and an integer K it is* NP*-hard to decide if in total at least K road sections can be identified.*

Proof. We perform a reduction from the NP-complete PLANAR MONOTONE 3-SAT problem [5]. An instance of PLANAR MONOTONE 3-SAT is a Boolean formula φ with n variables and m clauses (disjunctions of at most three literals) that satisfies the following additional requirements: (i) φ is *monotone*, i.e., every clause contains either only positive literals or only negative literals and (ii) the induced variable-clause graph H_φ of φ is planar and can be embedded in the plane with all variable vertices on a horizontal line, all positive clause vertices on one side of the line, all negative clauses on the other side of the line, and the edges drawn as rectilinear curves connecting clauses and contained variables on their respective side of the line. We construct a road map $\mathcal{M}_\varphi$ that mimics

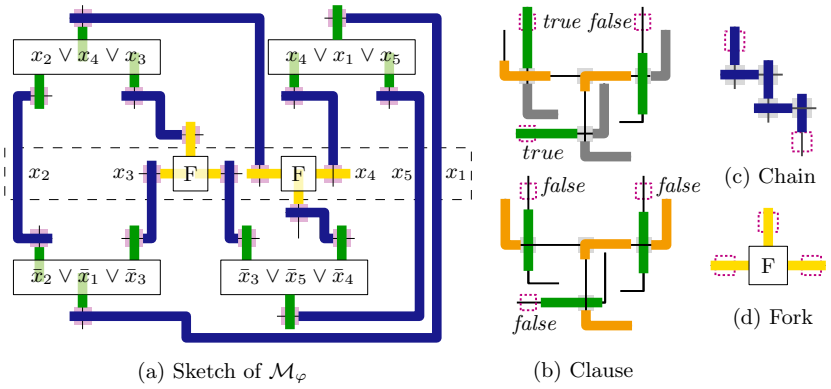

Fig. 2. Illustration of NP-hardness proof. (a) 3-Sat formula $\varphi = (x_4 \vee x_1 \vee x_5) \wedge (x_2 \vee x_4 \vee x_3) \wedge (\bar{x}_2 \vee \bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee \bar{x}_5 \vee \bar{x}_4)$ represented as road graph $\mathcal{M}_\varphi$. Truth assignment is $x_1 = \mathit{true}$, $x_2 = \mathit{true}$, $x_3 = \mathit{false}$, $x_4 = \mathit{false}$ and $x_5 = \mathit{false}$. (b) Clause gadget in two states. (c) The chain is the basic building block for the proof. (d) Schematized fork gadget.

the shape of the above embedding of H_φ by defining variable and clause gadgets, which simulate the assignment of truth values to variables and the evaluation of the clauses. We refer to Fig. 2 for a sketch of the construction.

Chain Gadget. The basic building block is the *chain gadget*, which consists of an alternating sequence of equally long horizontal and vertical roads with identical label

lengths that intersect their respective neighbors in the sequence and form junctions with them as indicated in Fig. 2c). Assume that the chain consists of $k \geq 3$ roads. Then each road except the first and last one decomposes into three road sections split by two junctions, a longer central section and two short end sections; the first and last road consist of only two road sections, a short one and a long one, separated by one junction. (These two roads will later be connected to other gadgets; indicated by dotted squares in Fig. 2c).) The label length and distance between junctions is chosen so that for each road either the central and one end section is identified, or no section at all is identified. For the first and last road, both sections are identified if the junction is covered and otherwise only the long section can be identified. We have k roads and $k-1$ junctions. Each label must block a junction, if it identifies two sections. So the best possible configuration blocks all junctions and identifies $2(k-1)+1 = 2k-1$ road sections.

The chain gadget has exactly two states, in which $2k-1$ road sections are identified. Either the label of the first road does not block a junction and identifies a single section and all subsequent roads have their label cover the junction with the preceding road in the sequence, or the label of the last road does not block a junction and all other roads have their label cover the junction with the successive road in the sequence. In any other configuration there is at least one road without any identified section and thus at most $2k-2$ sections are identified. We use the two optimal states of the gadget to represent and transmit the values *true* and *false* from one end to the other.

Fork Gadget. The *fork gadget* allows to split the value represented in one chain into two chains, which is needed to transmit the truth value of a variable into multiple clauses. To that end it connects to an end road of three chain gadgets by sharing junctions. The detailed description of the fork gadget is found in the full version [3].

Variable Gadget. We define the *variable gadgets* simply by connecting chain and fork gadgets into a connected component of intersecting roads. This construction already has the functionality of a variable gadget: it represents (in a labeling identifying the maximum number of road sections) the same truth value in all of its branches, synchronized by the fork gadgets, see the blue chains and yellow forks in Fig. 2a). More precisely, we place a sequence of chains linked by fork gadgets along the horizontal line on which the variable vertices are placed in the drawing H_φ. Each fork creates a branch of the variable gadget either above or below the line. We create as many branches above (below) the line as the variable has occurrences in positive (negative) clauses in φ. The first and last chain on the line also serve as branches. The synchronization of the different branches via the forks is such that either all top branches have their road labels pushed away from the line and all bottom branches pulled towards the line or vice versa. In the first case, we say that the variable is in the state *false* and in the latter case that it is in the state *true*. The example in Fig. 2 has two variables set to *true* and three variables set to *false*.

Clause Gadget. Finally, we need to create the clause gadget, which links three branches of different variables. The core of the gadget is a single road that consists of three sub-paths meeting in one junction. Each sub-path of that road shares another junction with one of the three incoming variable branches. Beyond each of these three junctions the final road sections are just long enough so that a label can be placed on the section. However, the section between the central junction of the clause road and the junctions with the literal roads is shorter than the label length. The road of the clause

gadget has six sections in total and we argue that the six sections can only be identified if at least one incoming literal evaluates to *true*. Otherwise at most five sections can be identified. By construction, each road in the chain of a false literal has its label pushed towards the clause, i.e., it blocks the junction with the clause road. As long as at least one of these three junctions is not blocked, all sections can be identified; see Fig. 2b). But if all three junctions are blocked, then only two of the three inner sections of the clause road can be identified and the third one remains unlabeled; see Fig. 2b).

Reduction. Obviously, the size of the instance $\mathcal{M}_\varphi$ is polynomial in n and m. If we have a satisfying variable assignment for φ, we can construct the corresponding road labeling and the number of identified road sections is six per clause and a fixed constant number K' of sections in the variable gadgets, i.e., at least $K = K' + 6m$. On the other hand, if we have a road labeling with at least K identified sections, each variable gadget is in one of its two maximum configurations and each clause road has at least one label that covers a junction with a literal road, meaning that the corresponding truth value assignment of the variables is indeed a satisfying one. This concludes the reduction.

Since MAXIDENTIFIEDROADS is an optimization problem, we only present the NP-hardness proof. Still, one can argue that the corresponding decision problem is NP-complete by guessing which junctions are covered by which label and then using linear programming for computing the label positions. We omit the technical details. Further, most roads in the reduction are paths, except for the central road in each clause gadget, which is a degree-3 star. In fact, we can strengthen Theorem 1 by using a more complex clause gadget instead that uses only paths; see full version [3].

4 An Efficient Algorithm for Tree-Shaped Road Maps

In this section we assume that the underlying road graph of the road map is a tree $T = (V, E)$. In Section 4.1 we present a polynomial-time algorithm to optimally solve MAXIDENTIFIEDROADS for trees; Section 4.2 shows how to improve its running time and space consumption. Our approach uses the basic idea that removing the vertices, whose embeddings lie in a curve $c \subseteq \mathsf{E}(T)$, splits the tree into independent parts. In particular this is true for labels. We assume that T is rooted at an arbitrary leaf ρ and that its edges are directed away from ρ; see Fig. 3. For two points $p, q \in \mathsf{E}(T)$ we define $\mathrm{d}(p, q)$ as the length of the shortest curve in $\mathsf{E}(T)$ that connects p and q. For two vertices u and v of T we also write $\mathrm{d}(u, v)$ instead of $\mathrm{d}(\mathsf{E}(u), \mathsf{E}(v))$. For a point $p \in E(T)$ we abbreviate the distance $\mathrm{d}(p, \rho)$ to the root ρ by d_p. For a curve ℓ in $\mathsf{E}(T)$, we call $p \in \ell$ the *lowest point* of ℓ if $\mathrm{d}_p \leq \mathrm{d}_q$, for any $q \in \ell$. As T is a tree, p is unique. We distinguish two types of curves in $\mathsf{E}(T)$. A curve ℓ is *vertical* if $h(\ell)$ or $t(\ell)$ is the lowest point of ℓ; otherwise we call ℓ *horizontal* (see

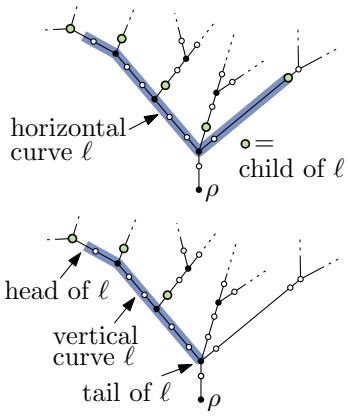

Fig. 3. Basic definitions.

Fig. 3). Without loss of generality we assume that the lowest point of each vertical curve ℓ is its tail $t(\ell)$. Since labels are modeled as curves, they are also either vertical or horizontal. For a vertex $u \in V$ let T_u denote the subtree rooted at u.

4.1 Basic Approach

We first determine a finite set of candidate positions for the heads and tails of labels, and transform T into a tree $T' = (V', E')$ by subdividing some of T's edges so that it contains a vertex for every candidate position. To that end we construct for each regular vertex $v \in V$ a chain of tightly packed vertical labels that starts at $\mathsf{E}(v)$, is directed towards ρ, and ends when either the road ends, or adding the next label does not increase the number of identified road sections. More specifically, we place a first vertical label ℓ_1 such that $h(\ell_1) = \mathsf{E}(v)$. For $i = 2, 3, \ldots$ we add a new vertical label ℓ_i with $h(\ell_i) = t(\ell_{i-1})$, as long as $h(\ell_i)$ and $t(\ell_i)$ do not lie on the same road section and none of ℓ_i's endpoints lie on a junction edge. We use the tails of all those labels to subdivide the tree T. Doing this for all regular vertices of T we obtain the tree T', which we call the *subdivision tree* of T. The vertices in $V' \setminus V$ are neither junction vertices nor regular vertices. Since each chain consists of $O(n)$ labels the cardinality of V' is $O(n^2)$. We call an optimal labeling $\mathcal{L}$ of T an *canonical labeling* if for each label $\ell \in \mathcal{L}'$ there exists a vertex v in T' with $\mathsf{E}(v) = h(\ell)$ or $\mathsf{E}(v) = t(\ell)$. The next lemma proves that is sufficient to consider canonical labelings.

Lemma 1. *For any road graph T that is a tree, there exists a canonical labeling $\mathcal{L}$.*

The idea behind the proof is to *push* the labels of an optimal labeling $\mathcal{L}$ as far as possible towards the leaves of T without changing the identified road sections; see Fig. 4. The labels then start or end either at a leaf, a regular vertex or at an endpoint of another label, which yields a canonical labeling. For the full proof see the full version [3].

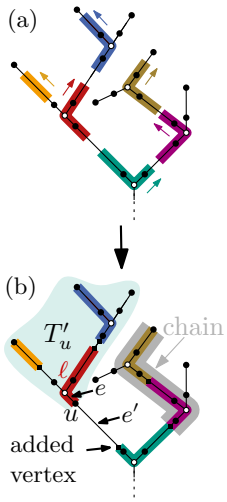

Fig. 4. Canonical labeling.

We now explain how to construct such a canonical labeling. To that end we first introduce some notations. For a vertex $u \in V'$ let $\mathcal{L}(u)$ denote a labeling that identifies a maximum number of road sections in T only using valid labels in $\mathsf{E}(T'_u)$, where T'_u denotes the subtree of T' rooted at u. Note that those labels also may identify the incoming road section of u, e.g., label ℓ in Fig. 4b) identifies the edge e'.

Further, the children of a vertex $u \in V'$ are denoted by the set $N(u)$; we explicitly exclude the parent of u from $N(u)$. Further, consider an arbitrary curve ℓ in $\mathsf{E}(T)$ and let $\ell' = \ell \setminus \{t(\ell), h(\ell)\}$. We observe that removing all vertices of T' contained in ℓ' together with their incident outgoing edges creates several independent subtrees. We call the roots of these subtrees (except the one containing ρ) *children* of ℓ (see Fig. 3). If no vertex of T' lies in ℓ', the curve is contained in a single edge $(u, v) \in E'$. In that case v is the only child of ℓ. We denote the set of all children of ℓ as $N(\ell)$.

For each vertex u in T' we introduce a set $C(u)$ of *candidates*, which model potential labels with lowest point $\mathsf{E}(u)$. If u is a regular vertex of T or $u \in V' \setminus V$, the set $C(u)$ contains all vertical labels ℓ with lowest point $\mathsf{E}(u)$. If u is a junction vertex, $C(u)$ contains all horizontal labels that start or end at a vertex of T' and whose lowest point is $\mathsf{E}(u)$. In both cases we assume that $C(u)$ also contains the degenerated curve $\perp_u = \mathsf{E}(u)$, which is the *dummy label* of u. We set $N(\perp_u) = N(u)$ and $\omega(\perp_u) = 0$.

For a curve ℓ we define $\mathcal{L}(\ell) = \bigcup_{v \in N(\ell)} \mathcal{L}(v) \cup \{\ell\}$. Thus, $\mathcal{L}(\ell)$ is a labeling comprising ℓ and the labels of its children's optimal labelings. We call a label $\overline{\ell} \in C(u)$ with $\overline{\ell} = \operatorname{argmax}\{\omega(\mathcal{L}(\ell)) \mid \ell \in C(u)\}$ an *optimal candidate* of u. Next, we prove that it is sufficient to consider optimal candidates to construct a canonical labeling.

Lemma 2. *Given a vertex u of T' and an optimal labeling $\mathcal{L}(u)$ and let $\overline{\ell}$ be an optimal candidate of u, then it is true that $\omega(\mathcal{L}(u)) = \omega(\mathcal{L}(\overline{\ell}))$.*

Proof. First note that $\omega(\mathcal{L}(u)) \geq \omega(\mathcal{L}(\overline{\ell}))$ because both labelings $\mathcal{L}(u)$ and $\mathcal{L}(\overline{\ell})$ only contain labels that are embedded in $\mathsf{E}(T'_u)$. By Lemma 1 we can assume without loss of generality that $\mathcal{L}(u)$ is a canonical labeling. Let ℓ be the label of $\mathcal{L}(u)$ with $\mathsf{E}(u)$ as the lowest point of ℓ (if it exists).

If ℓ exists, then the vertices in $N(\ell)$ are roots of independent subtrees, which directly yields $\omega(\mathcal{L}(u)) = \omega(\mathcal{L}(\ell))$. By construction of $C(u)$ we further know that ℓ is contained in $C(u)$. Hence, ℓ is an optimal candidate of u, which implies $\omega(\ell) = \omega(\overline{\ell})$.

If ℓ does not exist, then we have

$$\omega(\mathcal{L}(u)) = \omega\Big(\bigcup_{v \in N(u)} \mathcal{L}(v)\Big) \overset{(1)}{=} \omega\Big(\bigcup_{v \in N(\perp_u)} \mathcal{L}(v) \cup \{\perp_u\}\Big) = \omega(\mathcal{L}(\perp_u)).$$

Equality (1) follows from $N(\perp_u) = N(u)$ and the definition that $\perp_u$ does not identify any road section. Since $\perp_u$ is contained in $C(u)$, the dummy label $\perp_u$ is the optimal candidate $\overline{\ell}$. □

Algorithm 1 first constructs the subdivision tree $T' = (V', E')$ from T. Then starting with the leaves of T' and going to the root ρ of T', it computes an optimal candidate $\overline{\ell} =$`OptCandidate` (u) for each vertex $u \in V'$ in a bottom-up fashion. By Lemma 2 the labeling $\mathcal{L}(\overline{\ell})$ is an optimal labeling of T'_u. In particular $\mathcal{L}(\rho)$ is the optimal labeling of T.

Algorithm 1: Computing an optimal labeling of T.

Input: Road graph T, where T is a tree with root ρ.
Output: Optimal labeling $\mathcal{L}(\rho)$ of T.

```
T' ← compute subdivision tree of T
for each leaf v of T' do L(v) ← ∅
for each vertex u of T' considered in a bottom-up traversal of T' do
    L(u) ← L(OptCandidate(u))
return L(ρ)
```

Due to the size of the subdivision tree T' we consider $O(n^2)$ vertices. Implementing `OptCandidate`(u), which computes an optimal candidate $\overline{\ell}$ for u, naively, creates $C(u)$ explicitly. We observe that if u is a junction vertex, $C(u)$ may contain $O(n^2)$ labels; $O(n^2)$ pairs of road sections of different subtrees of u can be connected by horizontal labels. Each label can be constructed in $O(n)$ time using a breadth-first search. Thus, for each vertex u the procedure `OptCandidate` needs in a naive implementation $O(n^3)$ time, which yields $O(n^5)$ running time in total. Further, we need $O(n^2)$ storage to store T'. Note that we do not need to store $\mathcal{L}(u)$ for each vertex u of T', but by Lemma 2 we can reconstruct it using $\mathcal{L}(\overline{\ell})$, where $\overline{\ell}$ is the optimal candidate of u. To that end we store for each vertex of T' its optimal candidate $\overline{\ell}$ and $w(\mathcal{L}(\overline{\ell}))$.

Theorem 2. *For a road map with a tree as underlying road graph,* MAXIDENTIFIED-ROADS *can be solved in $O(n^5)$ time using $O(n^2)$ space.*

In case that all roads are paths, Algorithm 1 runs in $O(n^4)$ time, because for each $u \in V'$ the set $C(u)$ contains $O(n)$ labels. Further, besides the *primary objective* to identify a maximum number of road sections, Chirié [1] also suggested several additional *secondary objectives*, e.g., labels should overlap as few junctions as possible. Our approach allows us to easily incorporate secondary objectives by changing the weight function ω appropriately.

4.2 Improvements on Running Time

In this part we describe how the running time of Algorithm 1 can be improved to $O(n^3)$ time by speeding up `OptCandidate`(u) to $O(n)$ time.

For an edge $e = (u, v) \in E \cup E'$ we call a vertical curve $\ell \subseteq \mathsf{E}(T)$ an *e-rooted* curve, if $t(\ell) = \mathsf{E}(u)$, $h(\ell)$ lies on a road section, and $\mathrm{len}(\mathsf{E}(e) \cap \ell) = \min\{\mathrm{len}(\ell), \mathrm{len}(\mathsf{E}(e))\}$, i.e., ℓ emanates from $\mathsf{E}(u)$ passing through e; for example the red label in Fig. 4b) is an e-rooted curve. An e-rooted curve ℓ is *maximal* if there is no other e-rooted curve ℓ' with $\mathrm{len}(\ell) = \mathrm{len}(\ell')$ and $\omega(\mathcal{L}(\ell')) > \omega(\mathcal{L}(\ell))$. We observe that in any canonical labeling each vertical label ℓ is a (u, v)-rooted curve with $(u, v) \in E'$, and each horizontal label ℓ can be composed of a (u, v_1)-rooted curve ℓ_1 and a (u, v_2)-rooted curve ℓ_2 with $(u, v_1), (u, v_2) \in E'$ and $\mathsf{E}(u)$ is the lowest point of ℓ; see Fig. 6 and Fig. 7, respectively. Further, for a vertical curve c in $\mathsf{E}(T)$ its *distance interval* $I(c)$ is $[\mathrm{d}_{t(c)}, \mathrm{d}_{h(c)}]$. Since T is a tree, for every point p of c we have $\mathrm{d}_p \in I(c)$. Two vertical curves c and c' *superpose* each other if $I(c) \cap I(c') \neq \emptyset$; see Fig 5.

Next, we introduce a data structure that encodes for each edge (u, v) of T all maximal (u, v)-rooted curves as $O(n)$ superposition-free curves in $\mathsf{E}(T_u)$. In particular, each of those curves lies on a single road section such that all (u, v)-rooted curves ending on that curve are maximal and identify the same number of road sections. We define this data structure as follows.

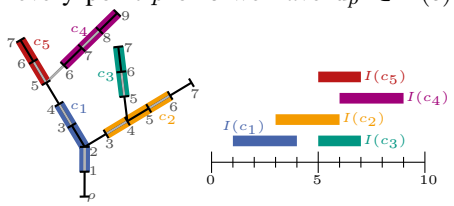

Fig. 5. Superposing curves, e.g., c_1 and c_2 superpose each other, while c_1 and c_5 do not. The tree is annotated with distance marks.

Definition 1 (Linearization). *Let $e = (u, v)$ be an edge of T. A tuple $(L, \overline{\omega})$ is called a* linearization *of e, if L is a set of superposition-free curves and $\overline{\omega}\colon L \to \mathbb{R}$ such that*
(1) for each curve $c \in L$ there is a road section e' in T_u with $c \subseteq \mathsf{E}(e')$,
(2) for each e-rooted curve ℓ there is a curve $c \in L$ with $\mathrm{len}(\ell) + \mathrm{d}_u \in I(c)$,
(3) for each point p of each curve $c \in L$ there is a maximal e-rooted curve ℓ with $h(\ell) = p$ and $\overline{\omega}(c) = \omega(\mathcal{L}(\ell))$.

Assume that we apply Algorithm 1 on T' and that we currently consider the vertex u of T'. Hence, we can assume that for each vertex $v \neq u$ of T'_u its optimal candidate and $\omega(\mathcal{L}(v))$ is given. We first explain how to speed up `OptCandidate` using linearizations. Afterwards, we present the construction of linearizations.

Application of linearizations. Here we assume that the linearizations are given for the edges of T. Concerning the type of u we describe how to compute its optimal candidate.

Case 1, u is regular. If u is a leaf, the set $C(u)$ contains only $\perp_u$. Hence, assume that u has one outgoing edge $e = (u, v) \in E'$, which belongs to a road R. Let P be the longest path of vertices in T'_u that starts at u and does not contain any junction vertex. Note that the path must be unique. Further, by construction of T' the last vertex w of P must be a regular vertex in V, but not in $V' \setminus V$. We consider two cases; see Fig 6.

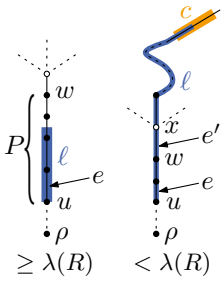

Fig. 6. Case 1

If $\mathrm{d}(u, w) \geq \lambda(R)$, the optimal candidate is either $\perp_u$ or the e-rooted curve ℓ of length $\lambda(R)$ that ends on $\mathsf{E}(P)$. By assumption and due to $\omega(\mathcal{L}(\perp_u)) = \omega(\mathcal{L}(v))$, we decide in $O(1)$ time whether $\omega(\mathcal{L}(\perp_u)) \geq \omega(\mathcal{L}(\ell))$, obtaining the optimal candidate.

If $\mathrm{d}(u, w) < \lambda(R)$, the optimal candidate is either $\perp_u$ or goes through a junction. Since w is regular, it has only one outgoing edge $e' = (w, x)$. Further, by the choice of P the edge e' is a junction edge in T; therefore the linearization $(L, \overline{\omega})$ of e' is given. In linear time we search for the curve $c \in L$ such that there is an e-rooted curve ℓ of length $\lambda(R)$ with its head on c. To that end we consider for each curve $c \in L$ its distance interval $I(c)$ and check whether there is $t \in I(c)$ with $t - \mathrm{d}_u = \lambda(R)$. Note that using a binary search tree for finding c speeds this procedure up to $O(\log n)$ time, however, this does not asymptotically improve the total running time. The e-rooted curve ℓ then can be easily constructed in $O(n)$ time by walking from c to u in $\mathsf{E}(T)$.

If such a curve c exist, by definition of a linearization the optimal candidate is either $\perp_u$ or ℓ, which we can decide in $O(1)$ time by checking $\omega(\mathcal{L}(\perp_u)) \geq \omega(\mathcal{L}(\ell))$. Note that we have $\omega(\mathcal{L}(\perp_u)) = \omega(\mathcal{L}(v))$ and $\omega(\mathcal{L}(\ell)) = \overline{\omega}(c)$. If c does not exist, again by definition of a linearization there is no vertical label $\ell \in C(u)$ and $\perp_u$ is the optimal candidate.

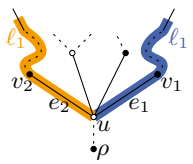

Fig. 7. Case 2

Case 2, u is a junction vertex. The set $C(u)$ contains horizontal labels. Let ℓ be such a label and let $e_1 = (u, v_1)$ and $e_2 = (u, v_2)$ be two junction edges in E covered by ℓ; see Fig. 7. Then there is an e_1-rooted curve ℓ_1 and an e_2-rooted curve ℓ_2 whose composition is ℓ. Further, we have $\omega(\mathcal{L}(\ell)) = \omega(\mathcal{L}(\ell_1) \cup \mathcal{L}(\ell_2)) + \sum_{v \in N(u) \setminus \{v_1, v_2\}} \omega(\mathcal{L}(v))$. We use this as follows.

Let e_1 and e_2 be two outgoing edges of u that belong to the same road R, and let $(L_1, \overline{\omega}_1)$ and $(L_2, \overline{\omega}_2)$ be the linearizations of e_1 and e_2, respectively. We define for e_1 and e_2 and their linearizations the operation $\text{opt-cand}(L_1, L_2)$ that finds an optimal candidate of u restricted to labels identifying e_1 and e_2.

For $i = 1, 2$ let $d_i = \max\{\mathrm{d}_u \mid u \text{ is vertex of } T_{v_i}\}$ and let $f_u(t) = \mathrm{d}_u - (t - \mathrm{d}_u) = 2\,\mathrm{d}_u - t$ be the function that "mirrors" the point $t \in \mathbb{R}^2$ at d_u. Applying $f_u(t)$ on the boundaries of the distance intervals of the curves in L_1, we first mirror these intervals such that they are contained in the interval $[2\,\mathrm{d}_u - d_1, \mathrm{d}_u]$; see Fig. 8. Thus, the curves in $L_1 \cup L_2$ are mutually superposition-free such that their distance intervals lie in $J = [2\,\mathrm{d}_u - d_1, d_2]$.

We call an interval $[x, y] \subseteq J$ a *window*, if it has length $\lambda(R)$, $\mathrm{d}_u \in [x, y]$ and there are curves $c_1 \in L_1$ and $c_2 \in L_2$ with $x \in I(c_1)$ and $y \in I(c_2)$; see Fig. 8. By the definition of a linearization there is a maximal e_1-rooted curve ℓ_1 ending on c_1 and a maximal e_2-rooted curve ℓ_2 ending on c_2 such that $\text{len}(\ell_1) + \text{len}(\ell_2) = \lambda(R)$. Consequently, the composition of ℓ_1 and ℓ_2 forms a horizontal label ℓ with $\omega(\mathcal{L}(\ell)) = \omega(\mathcal{L}(\ell_1) \cup \mathcal{L}(\ell_2)) + \sum_{v \in N(u) \setminus \{v_1, v_2\}} \mathcal{L}(v)$; we call $\omega(\mathcal{L}(\ell))$ the *value* of the window. Using a simple sweep from left to right we compute for the distance interval $I(c)$ of each curve $c \in L_1 \cup L_2$ a window $[x, y]$ that starts or ends in $I(c)$ (if such a window exists). The result of $\text{opt-cand}(L_1, L_2)$ is then the label ℓ of the window with maximum value. For each pair e_1 and e_2 of outgoing edges we apply $\text{opt-cand}(L_1, L_2)$ computing a label ℓ. By construction either the label ℓ with maximum $\omega(\ell)$ or $\perp_u$ is the optimal candidate for u, which we can check in $O(1)$ time. Later on we prove that we consider only linearizations of linear size. Since each vertex of T' has constant degree, we obtain the next lemma.

Lemma 3. *For each $u \in V'$ the optimal candidate can be found in $O(n)$ time.*

Construction of linearizations. It remains to show that a linearization of an edge $e = (u, v)$ can be constructed in $O(n)$ time assuming that the linearizations $(L_1, \overline{\omega}_1), \ldots, (L_k, \overline{\omega}_k)$ of the outgoing edges $e_1 = (v, w_1), \ldots, e_k = (v, w_k)$ of v are given. Due to space restrictions we only give a rough sketch; the details can be found in the full version [3]. For $1 \le i \le k$ let T_i be the tree induced by the edges e, e_i and the edges of T_{w_i}. We first extend each linearization $(L_i, \overline{\omega}_i)$ to a linearization restricted to the tree T_i, i.e., conceptually we assume that T_u only consists of the edges of T_i. To that end we basically extend L_i by disjoint sub-curves of $\mathsf{E}(e)$ if e is a road section and update $\overline{\omega}_i$. Afterwards we merge those constructed linearizations to one linearization $(L, \overline{\omega})$ of e without any restrictions on T_u. In particular

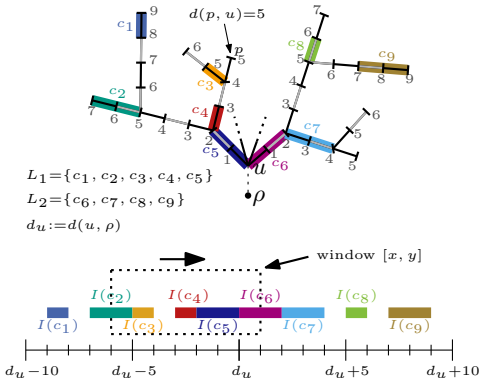

Fig. 8. Constructing the optimal candidate of u based on the linearizations $(L_1, \overline{\omega}_1)$ and $(L_2, \overline{\omega}_2)$. The tree is annotated with distance marks.

we enforce that L is a set of superposition-free curves, which we achieve by splitting superposing curves c and c' into three superposition-free curves c_1, c_2, c_3 such that each of them is either contained in c or c' and $\bigcup_{i=1}^{3} I(c_i) = I(c) \cup I(c')$. The choice of c_1, c_2 and c_3 depends on the number of road sections identified by an e-rooted curve that ends on c and c', respectively. We prove that this merging runs in $O(n)$ time per edge $e \in E$ and $(L, \overline{\omega})$ has size $O(n)$. This and Lemma 3 yield the next proposition.

Proposition 1. *For a road map $\mathcal{M}$ with a tree T as underlying road graph,* MAXIDENTIFIEDROADS *can be solved in $O(n^3)$ time.*

Since T' contains $O(n^2)$ vertices, the algorithm needs $O(n^2)$ space. This can be improved to $O(n)$ space. To that end T' is constructed *on the fly* while executing Algorithm 1. Parts of T' that become unnecessary are discarded. In the full version [3] we prove that it is sufficient to store $O(n)$ vertices of T' at any time such that the optimal labeling can still be constructed. We summarize in the following theorem.

Theorem 3. *For a road map $\mathcal{M}$ with a tree T as underlying road graph,* MAXIDENTIFIEDROADS *can be solved in $O(n^3)$ time using $O(n)$ space.*

5 Conclusions and Outlook

In this paper we investigated the problem of maximizing the number of identified road sections in a labeling of a road map; we showed that it is NP-hard in general, but can be solved in $O(n^3)$ time and linear space for the special case of trees.

The underlying road graphs of real-world road maps are rarely trees. Initial experimental evidence indicates, however, that road maps can be decomposed into a large number of subgraphs by placing trivially optimal road labels and removing the corresponding edges from the graph. It turns out that between 85.1% and 97.7% of the resulting subgraphs are actually trees, which we can label optimally by our proposed algorithm. As a consequence, this means that a large fraction (between 88.6% and 96.1%) of all road sections in our real-world road graphs can be labeled optimally by combining this simple preprocessing strategy with the tree labeling algorithm. We are investigating further heuristic and exact approaches for labeling the remaining non-tree subgraphs (e.g., by finding suitable spanning trees and forests) for a separate companion paper.

References

1. F. Chirié. Automated name placement with high cartographic quality: City street maps. *Cartography and Geo. Inf. Science*, 27(2):101–110, 2000.
2. S. Edmondson, J. Christensen, J. Marks, and S. M. Shieber. A general cartographic labelling algorithm. *Cartographica*, 33(4):13–24, 1996.
3. A. Gemsa, B. Niedermann, and M. Nöllenburg. Label placement in road maps. *CoRR*, abs/1501.07188, 2015.
4. E. Imhof. Positioning names on maps. *Amer. Cartogr.*, pages 128–144, 1975.
5. D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.
6. S. Maass and J. Döllner. Embedded labels for line features in interactive 3d virtual environments. In *Proc. 5th Int. Conf. Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, AFRIGRAPH '07, pages 53–59. ACM, 2007.
7. G. Neyer and F. Wagner. Labeling downtown. In *Algorithms and Complexity (CIAC'00)*, volume 1767 of *LNCS*, pages 113–124. Springer, 2000.
8. A. Reimer and M. Rylov. Point-feature lettering of high cartographic quality: A multi-criteria model with practical implementation. In *EuroCG'14*, Ein-Gedi, Israel, 2014.
9. S. Seibert and W. Unger. The hardness of placing street names in a Manhattan type map. *Theor. Comp. Sci.*, 285:89–99, 2002.
10. T. Strijk. *Geometric Algorithms for Cartographic Label Placement*. Dissertation, Utrecht University, 2001.
11. M. Vaaraniemi, M. Treib, and R. Westermann. Temporally coherent real-time labeling of dynamic scenes. In *Proc. 3rd Int. Conf. Comput. Geospatial Research Appl.*, COM.Geo '12, pages 17:1–17:10. ACM, 2012.
12. M. van Kreveld. Geographic information systems. In *Handbook of Discrete and Computational Geometry, Second Edition*, chapter 58, pages 1293–1314. CRC Press, 2010.
13. A. Wolff, L. Knipping, M. van Kreveld, T. Strijk, and P. K. Agarwal. A simple and efficient algorithm for high-quality line labeling. In *Innovations in GIS VII: GeoComputation*, chapter 11, pages 147–159. Taylor & Francis, 2000.
14. A. Wolff and T. Strijk. The map labeling bibliography. `http://liinwww.ira.uka.de/bibliography/Theory/map.labeling.html`, 2009.