Algorithmica (2020) 82:1881–1908
https://doi.org/10.1007/s00453-020-00678-7

Placing Labels in Road Maps: Algorithms and Complexity

Andreas Gemsa[1] · Benjamin Niedermann[2] · Martin Nöllenburg[3]

Received: 21 June 2018 / Accepted: 13 January 2020 / Published online: 1 February 2020

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. In this paper, we consider the computational 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 $O(n^3)$ time if the road map is an embedded tree with n vertices and constant maximum degree. 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 maps.

Keywords Map labeling · Road maps · NP-hardness · Efficient tree-based algorithm

1 Introduction

Map labeling is a well-known cartographic application problem in computational geometry [13, 16]. 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

A preliminary version of this paper has appeared as follows. *Label Placement in Road Maps. In: Proceedings of the 9th Conference on Algorithms and Complexity (CIAC'15), volume 9079 of Lecture Notes in Computer Science, pp. 221–234. Springer, 2015.*

✉ Benjamin Niedermann
niedermann@uni-bonn.de

Martin Nöllenburg
noellenburg@ac.tuwien.ac.at

[1] Karlsruhe Institute of Technology, Karlsruhe, Germany

[2] University of Bonn, Bonn, Germany

[3] TU Wien, Vienna, Austria

considerably less attention. In this paper we address labeling line features, namely roads in a road map.

Geometrically speaking, a *road map* is the representation of a *road graph* G as an arrangement of fat curves, i.e., curves with fixed, positive stroke width, 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 all road sections are identified unambiguously but at the same time the road map is not overloaded with labels.

Chirié [1] presented a set of rules and quality criteria for label placement in road maps based on interviews with cartographers. These include 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 [3]. 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. [15] introduced an algorithm for a 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 [11] considered grid-shaped road networks. They showed that in those networks it is NP-complete to decide whether for every road at least one label can be placed without any overlaps between labels. Further, they prove that the according optimization problem maximizing the number of roads having at least one label is APX-hard. Yet, Neyer and Wagner [7] introduced a practically efficient algorithm that finds such a grid labeling if possible. Maass and Döllner [5] 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. [14] used a force-based labeling algorithm for 2D and 3D scenes including road label placement. Schwartges et al. [12] presented a simple heuristic for maximizing the number of placed road labels in interactive maps providing operations for panning, zooming and rotating the map. Finally, Schwartges et al. [9] considered the problem of labeling roads in 3-dimensional maps. In order to avoid distortion of the labels by projecting them onto the road, they used *billboards* instead, i.e., labels connected to their roads by short leaders.

Contribution and Outline While it is sufficient to place a single label per road to clearly identify all its road sections in grid-shaped road networks, this is not the case in general road networks. Consider the schematic example in Fig. 1. In Fig. 1a, it is not obvious whether the road section in the center belongs to *Knuth St.* or to *Turing St.* On the other hand, simply maximizing the number of placed labels, as often done

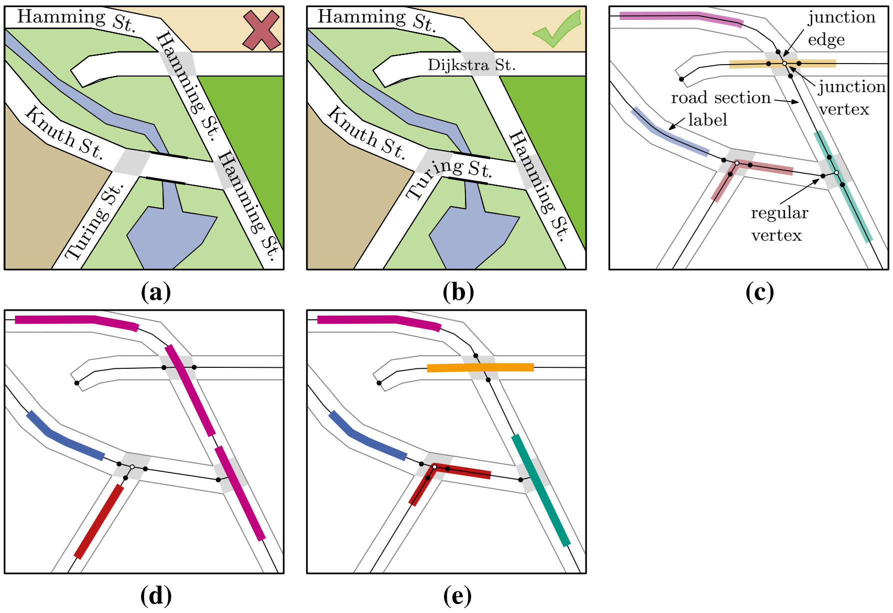


Fig. 1 **a**, **b** Two ways to label the same road network. Junctions are marked gray. **b** Labels all road sections. **c** Illustration of the road graph and relevant terms. **d** Graph representation of (**a**) with five labels and five labeled road sections. **e** Graph representation of (**b**) with five labels and eight labeled road sections, which is optimal with respect to the considered objective (Color figure online)

for labeling point features, can cause undesired effects like unnamed roads or clumsy label placements (e.g., the unlabeled *Dijkstra St.* and the threefold labeled *Hamming St.* in Fig. 1a. In order to overcome the problems outlined, we aim for maximizing *the number of labeled road sections*, i.e., the number of road sections that can be clearly assigned to labels; see Fig. 1b. Further, in contrast to the approaches of Chirié [1] and Strijk [10, Ch. 9], we consider label placement in road maps globally for the entire map, applying a continuous sliding model. More precisely, as the underlying model, we introduce a new and versatile planar graph model based on criteria (C1)–(C3); see Sect. 2.

We take an algorithmic, mathematical perspective on the optimization problem of maximizing the number of labeled road sections. In Sect. 3 we show that the problem of maximizing the number of labeled road sections is NP-hard for general road graphs, even if every road is a path. For the special case that the road graph is a tree, we present a polynomial-time algorithm in Sect. 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, as we demonstrated in an applied companion paper [6].

2 Model

As argued before, 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, by being contained inside the road curves, 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. We use these observations to define the following abstract road graph model.[1]

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* consists of a junction vertex v and its incident junction edges. 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 that no two road sections in G are incident to the same (regular) 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 passing through a single 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 further assume that each road $R \in \mathcal{R}$ has a name (to be displayed in its label) whose length we denote by $\lambda(R)$.

For simplicity, we identify the embedding $\mathsf{E}(G)$ with the subset of 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 $\ell : [0,1] \to \mathsf{E}(G)$. 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 $\ell(0)$ of ℓ is denoted as the *head* $h(\ell)$ and the endpoint $\ell(1)$ as the *tail* $t(\ell)$. The length of ℓ is denoted by $\operatorname{length}(\ell)$. The curve ℓ (partly) *covers* a road section r if $\ell \cap \mathsf{E}(r) \neq \emptyset$. In that case we say that ℓ *labels* the road section r. For a set $\mathcal{L}$ of curves, $\omega(\mathcal{L})$ is the number of road sections that are labeled 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 label the same road section twice.

[1] Our companion paper [6] explains how to extract an abstract road graph from a given road network such that the abstract model can be applied in practice.

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 labeled road sections, i.e., for any labeling $\mathcal{L}'$ we have $\omega(\mathcal{L}') \leq \omega(\mathcal{L})$. We call this algorithmic optimization problem MaxLabeledRoads. While Fig. 1d shows a sub-optimal labeling, Fig. 1e shows an optimal labeling of the same road map with respect to the considered objective function.

Note that assuming the road graph G to be planar is not a restriction in practice [6]. 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' into shorter 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.

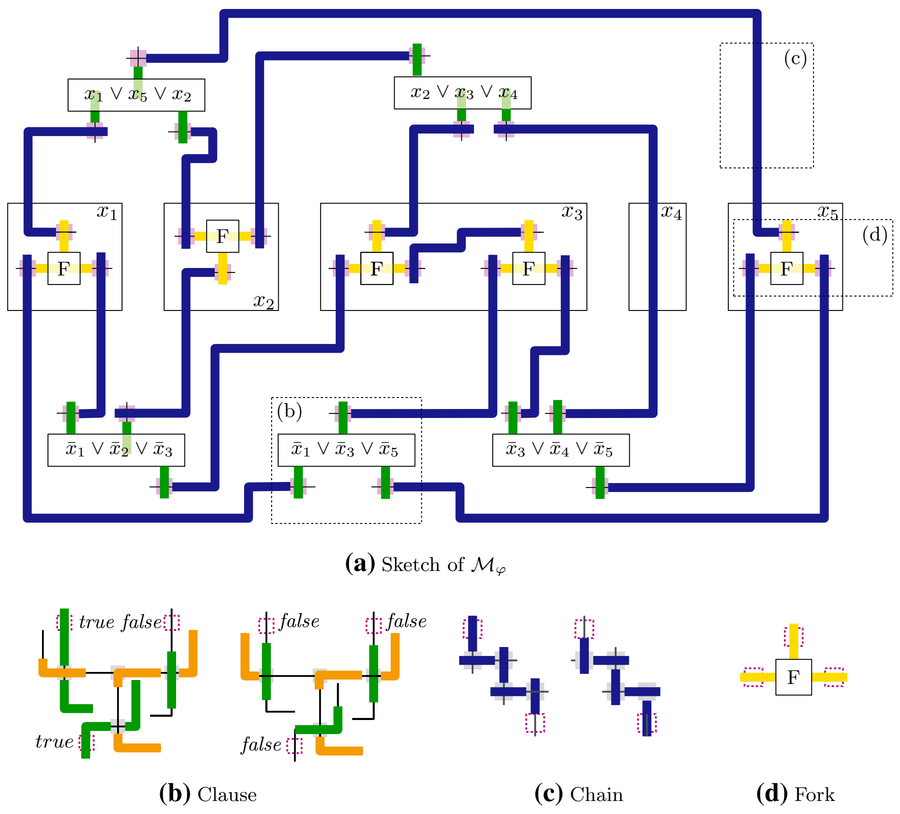
3 Computational Complexity

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

Theorem 1 *For a given road map $\mathcal{M}$ and an integer K it is* NP*-complete to decide if in total at least K road sections can be labeled even if $\mathcal{M}$ has maximum degree 4.*

Proof One can argue that the decision problem is in NP by guessing which junctions are covered by which label and then using linear programming for computing the label positions. More precisely, by means of an integer linear programming formulation with binary and continuous variables one can describe MaxLabeledRoads [6]. After guessing all binary variables contained in this formulation, a linear programming formulation remains, which can be solved in polynomial time. For the technical details of that formulation we refer to the companion paper [6].

We perform a reduction from the NP-complete planar monotone 3-Sat problem [4]. 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.



$x_1 \vee x_5 \vee x_2$
$x_2 \vee x_3 \vee x_4$
x_1
x_2
x_3
x_4
x_5
F
(c)
(d)
(b)
$\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3$
$\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_5$
$\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5$
(a) Sketch of $\mathcal{M}_\varphi$
true
false
true
false
false
false
(b) Clause
(c) Chain
(d) Fork

Fig. 2 Illustration of the NP-hardness proof. **a** 3-Sat formula $\varphi = (x_1 \vee x_5 \vee x_2) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_5) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5)$ represented as road graph $\mathcal{M}_\varphi$. The gadgets are illustrated schematically and the chain gadgets are represented as polylines without showing each single road. The truth assignment is $x_1 = false$, $x_2 = true$, $x_3 = false$, $x_4 = false$ and $x_5 = false$. **b** Clause gadget in two states. The green labels are the connectors shown in (**a**). The orange labels are placed depending on the state of the clause. **c** The chain is the basic building block for the proof. The illustration shows both possible states labeling the maximum number of road sections. **d** Schematized fork gadget. For details see also Fig. 3 (Color figure online)

We construct a road map $\mathcal{M}_\varphi$ that mimics 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

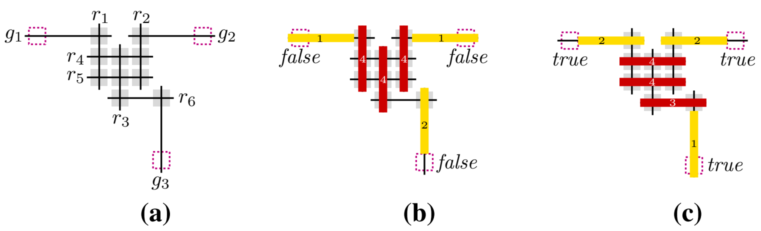


Fig. 3 Illustration of the fork gadget. The dotted rectangles (lilac) indicate the parts of the road sections that are later used to connect the gadgets with other gadgets. For each label the number of labeled road sections is given. Each road section is labeled by at most one label. **a** Structure of the fork gadget. **c** Configuration transmitting the value *false*. **b** Configuration transmitting the value *true* (Color figure online)

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 labeled, or no section at all is labeled. For the first and last road, both sections are labeled if the junction is covered and otherwise only the long section can be labeled. We have k roads and $k-1$ junctions. Each label must block a junction if it labels two sections. So the best possible configuration blocks all junctions and labels $2(k-1)+1=2k-1$ road sections.

The chain gadget has exactly two states in which $2k-1$ road sections are labeled. Either the label of the first road does not block a junction and labels 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 labeled section and thus at most $2k-2$ sections are labeled. 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 us 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. It connects to the end roads of three chain gadgets by sharing junctions.

The core of the fork, as illustrated in Fig. 3, consists of six roads $r_1, \dots, r_6$, where r_1, r_2, and r_3 are vertical line segments and r_4, r_5, and r_6 are horizontal line segments. We arrange these roads such that r_1 and r_2 each have one junction with r_4 and one junction with r_5. Further, r_3 has one junction with r_4, one with r_5 and one with r_6. The label length of these roads is chosen so that it is exactly the length of the roads. Hence, a placed label blocks all road sections of the roads.

Further, there are three roads g_1, g_2, g_3 such that g_1 has one junction with r_1, g_2 has one junction with r_2, and g_3 has one junction with r_6. In all three cases we place the junction so that it splits the road into a short road section that is shorter than the road's label length and a long road section that has exactly the road's label length. We call g_1, g_2 and g_3 *gates*, because later these roads will be connected by junctions to the end roads of chains. To that end, these *connecting* junctions will be placed on the long road sections of the gates; see the purple dotted areas in Fig. 3.

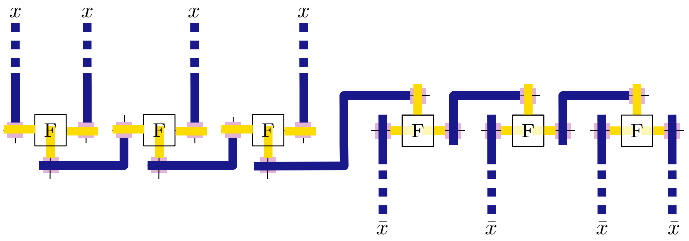


Fig. 4 Illustration of the variable gadget. The gadget has four connections for the positive literal x and four connections for the negative literal $\bar{x}$. In the current state the variable x is set to *false* (Color figure online)

The fork gadget has exactly two states, in which 16 road sections are labeled.[2] In the first state the labels of r_1, r_2 and r_3 are placed; see Fig. 3b. Hence, the labels of g_1 and g_2 label only the long road sections of g_1 and g_2, but not the short ones. The label of g_3 labels both the long and short road section of g_3. In the second state the labels of r_4, r_5, r_6 are placed; see Fig. 3c. Hence, the labels of g_1 and g_2 label the long and short road sections of g_1 and g_2, while only the long road section of g_3 is labeled. In any other configuration fewer road sections are labeled. We use the two optimal states of the gadget to represent and transmit the values *true* and *false* from one gate to the other two gates. More specifically the gates g_1 and g_2 are connected with chains that lead to the same literal, while g_3 is connected with a chain that leads to the complementary literal.

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 that labels 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 Figs. 2 and 4a. 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 leftmost and rightmost 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 one variable set to *true*, namely x_2, and four variables set to *false*, namely x_1, x_3, x_4 and x_5. Note that in case a variable occurs only in one positive and one negative clause (see x_4 in Fig. 2a), we connect the corresponding literals via a single chain without introducing forks for that variable.

[2] For counting we consider the fork gadgets without connected chain gadgets, i.e., the roads g_1, g_2, and g_3 consist of two road sections and two junction edges each. Connecting chain gadgets to the fork gadgets do not change their functionality and the counting argument.

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 subpaths meeting in one junction. Each subpath 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 labeled if at least one incoming literal evaluates to *true*. Otherwise at most five sections can be labeled. 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 labeled; see Fig. 2b. But if all three junctions are blocked, then only two of the three inner sections of the clause road can be labeled and the third one remains unlabeled. We observe that this requires that the core of the gadget is a single road.

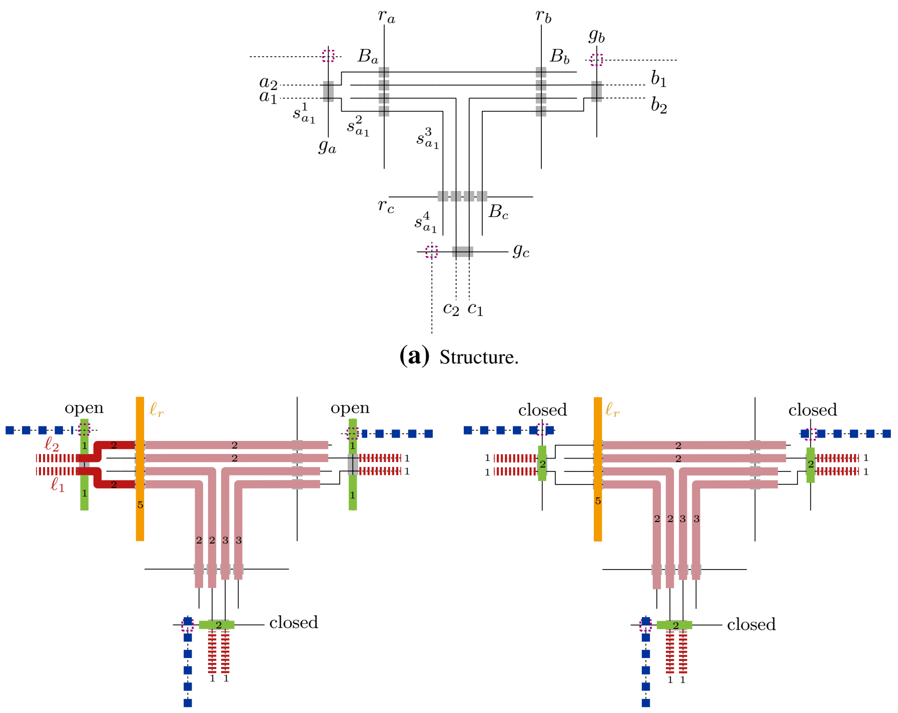
Reduction Obviously, the size of the instance $\mathcal{M}_\varphi$ is polynomial in n and m. If we have a satisfying variable assignment for φ, we can construct the corresponding road labeling and the number of labeled road sections is six per clause and a fixed constant number K' of sections in the variable gadgets, i.e., at least $K = K' + 6m$. On the other hand, if we have a road labeling with at least K labeled 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. □

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 consists only of paths as described next.

Theorem 2 *For a given road map $\mathcal{M}$ and an integer K it is* NP*-complete to decide if in total at least K road sections can be labeled, even if $\mathcal{M}$ has maximum degree 4 and all roads are paths.*

Proof The clause gadget consists of twelve roads, r_a, r_b, r_c, g_a, g_b, g_c, a_i, b_i, and c_i with $i \in \{1, 2\}$ that all are paths; see Fig. 5. We choose r_a, r_b, and r_c such that r_a and r_b consist of vertical segments and r_c consists of horizontal segments. Going along r_a from bottom to top, the junctions with the roads a_1, a_2, c_1 and b_1 occur in the sequence $B_a = (a_1, c_2, b_1, a_2)$. Going along r_b from top to bottom, the junctions with the roads b_1, b_2, a_2 and c_1 occur in the sequence $B_b = (a_2, b_1, c_1, b_2)$. Going along r_c from right to left, the roads b_2, c_1, c_2 and a_1 occur in the sequence $B_c = (b_2, c_1, c_2, a_1)$. Further, the label lengths of r_a, r_b, and r_c is chosen such that exactly one label can be placed.

We now describe junctions of the roads g_a, g_b, g_c, a_i, b_i, and c_i with $i \in \{1, 2\}$. The road a_1 first intersects g_a, then r_a, and finally r_c. Let $s^1_{a_1}$, $s^2_{a_1}$, $s^3_{a_1}$ and $s^4_{a_1}$ denote



(a) Structure.

(b) One literal is *true*.

(c) All literals are *false*.

Fig. 5 Illustration of an alternative clause gadget that consists only of paths as roads. The dotted rectangles (lilac) indicate the parts of the road sections that are used to connect the gadgets with other gadgets. For each label the number of labeled road sections is given. Each road section is labeled by at most one label. **a** Structure of the clause gadget. **b** Optimal labeling for the case that at least one literal is *true*. **c** Optimal labeling for the case that all literals are *false* (Color figure online)

these road sections in that particular order. The length of $s^1_{a_1}$ is chosen such that a single label can be placed on $s^1_{a_1}$, while the others are shorter than the label length of a_1. More specifically, we define a_1's label length such that a label covers the sections in either $\{s^1_{a_1}\}$, $\{s^1_{a_1}, s^2_{a_1}\}$, $\{s^1_{a_1}, s^2_{a_1}, s^3_{a_1}\}$, $\{s^2_{a_1}, s^3_{a_1}, s^4_{a_1}\}$ or $\{s^3_{a_1}, s^4_{a_1}\}$. We define the intersections and the label length for a_2 analogously. Further, g_a intersects a_1 and a_2 in one junction, i.e., the edge of g_a connecting both junction vertices is a junction edge. The label length of g_a is chosen so that a label can cross g_a's only junction. The length of g_a's road sections is at least as long as g_a's label length. We call g_a a *gate*, because later this road will be connected to the end road of a chain by a junction; see the lilac dotted square in Fig. 5a. For b_1, b_2, c_1, c_2 we introduce analogous junctions and road sections, however, b_1 and b_2 intersect g_b instead of g_a, and c_1 and c_2 intersect g_c instead of g_a.

In order to label both road sections of a gate, either two labels can be placed on the road sections separately, or one label that goes through the junction. In the former case the gate is *open*, and in the latter case it is *closed*; see Fig. 5b. We observe that it only makes sense to close a gate if at least one road section of the gate does

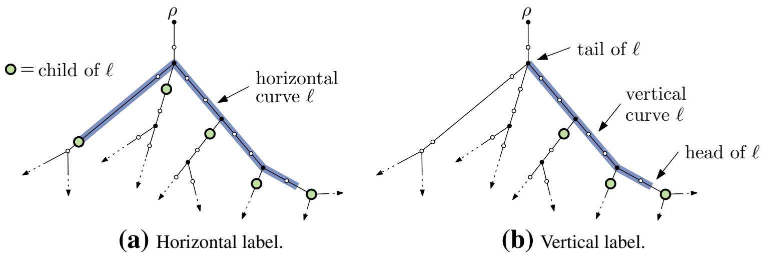


Fig. 6 Basic definitions of horizontal and vertical labels

not allow to place a label that is only contained in that road section. This case will occur if and only if the connected chain transmits the value *false* to the clause.

Assume that at least one gate is open, i.e., one literal of the clause is true; see Fig. 5b for an example with two open gates. Without loss of generality let g_a be open. We place a label ℓ_r on r_a such that it crosses the junctions of sequence B_a and labels five sections. Since g_a is open, we can place a label ℓ_1 that labels $s^1_{a_1}$ and $s^2_{a_1}$. Analogously, we can place a label ℓ_2 labeling $s^1_{a_2}$ and $s^2_{a_2}$. Placing further labels as indicated in Fig. 5b, we label five road sections of r_a and all road sections of any other road except for $s^4_{c_2}$, $s^4_{b_1}$. Hence, 33 road sections are labeled.

We observe that we can place the labels of b_1, b_2, c_1, c_2 such that they do not cross the junctions of g_b and g_c, respectively. Hence, it does not matter whether g_b and g_c are closed or open, i.e., it does not matter whether the corresponding literals are *true* or *false*.

We now argue that this is an optimal labeling. If $s^4_{c_2}$ or $s^4_{b_1}$ were labeled, the label ℓ_r must be placed such that the junctions of r with c_2 and b_1 are not crossed, respectively. This decreases the number of labeled road sections at least as much as labeling $s^4_{c_2}$ and $s^4_{b_1}$ increases the number of labeled road sections. In order to label at least one of the unlabeled road sections of r, we need to place a label that crosses B_b or B_c. Obviously, this yields a smaller number of labeled road sections than 31.

Finally, assume that all gates are closed as in Fig. 5c. Consider the same labeling as before. This time, however, we cannot label $s^2_{a_1}$ and $s^2_{a_2}$ anymore. Hence, this labeling has only 31 labeled road sections. Obviously, it cannot be improved by changing the placement of the remaining labels or adding more labels. □

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 Sect. 4.1 we present a polynomial-time algorithm to optimally solve MaxLabeledRoads for trees; Sect. 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

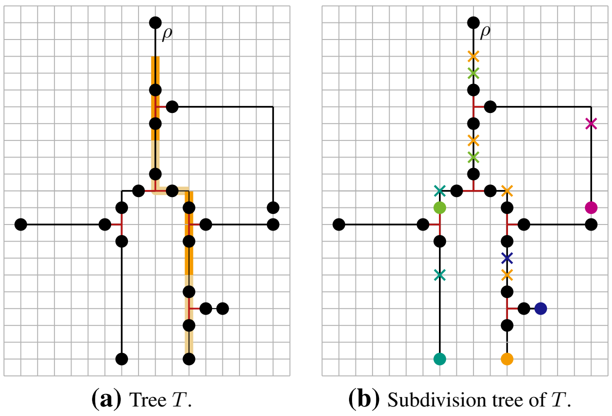


Fig. 7 Illustration of subdivision trees. **a** The original tree T with root ρ and a tightly packed chain of labels (orange) starting at a leaf of T. The label length is five. **b** The corresponding subdivision tree. The inserted vertices (marked by $\times$) have the same color as the original vertex at which the according chain of tightly packed labels has started. In case that two subdivision vertices coincide only one is kept (Color figure online)

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. 6. 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 $p \in \mathsf{E}(T)$ 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 *highest 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 highest point of ℓ; otherwise we call ℓ *horizontal*; see Fig. 6. Without loss of generality we assume that the highest 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 the resulting tree 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 labeled road sections; see Fig. 7. More specifically, we place a first vertical label ℓ_1 such that $h(\ell_1) = \mathsf{E}(v)$; recall that $t(\ell_1)$ is the highest point of ℓ_1. 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 lies on a junction edge. We use the tails of all these 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 $V' \backslash V$ are neither junction vertices nor regular

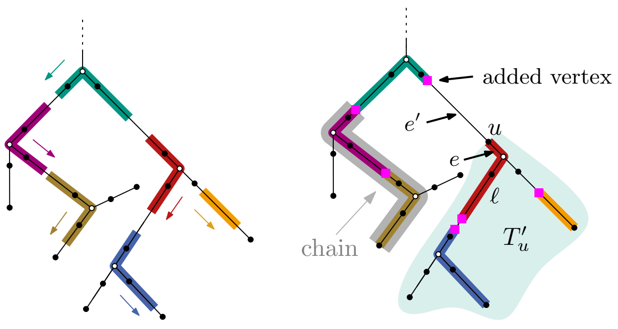


Fig. 8 Illustration of canonical labelings. **a** Each label is moved away from the root as far as possible while its head and tail must remain on their respective road sections. **b** The canonical labeling obtained from (**a**). The tree is subdivided by additional vertices (pink squares) at the tails and heads of the labels (Color figure online)

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 a *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 it is sufficient to consider canonical labelings.

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

Proof Let $\mathcal{L}$ be an optimal labeling of T. We *push* the labels of $\mathcal{L}$ as far as possible towards the leaves of T without changing the labeled road sections; see Fig. 8. More specifically, starting with the labels closest to the leaves, we move each label away from the root as far as possible while its head and tail must remain on their respective road sections. For a vertical label this direction is unique, while for horizontal labels we can choose any of the two possible directions. Then, for each label its head or tail either coincides with a leaf of T, with some internal regular vertex, or with the head of another label. Consequently, each vertical label belongs to a chain of tightly packed vertical labels starting at a regular vertex $v \in V$. Further, the head or tail of each horizontal label coincides with the end of a chain of tightly packed vertical labels or with a regular vertex of T, which proves the claim. □

We now explain how to construct a canonical labeling that is optimal among all possible labelings. To that end, we first introduce some notations. For a vertex $u \in V'$ let $\mathcal{L}(u)$ denote a labeling that labels a maximum number of road sections in T using only valid labels in $\mathsf{E}(T'_u)$, where T'_u denotes the subtree of T' rooted at u. Note that these labels may also cover the incoming road section of u, e.g., label ℓ in Fig. 8b covers 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. 6). 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 highest 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 highest 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 highest point is $\mathsf{E}(u)$. In both cases we assume that $C(u)$ also contains the degenerate 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 *Let u be a vertex of T' with 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 $\omega(\mathcal{L}(u)) \geq \omega(\mathcal{L}(\overline{\ell}))$ by the definition of $\mathcal{L}(u)$ and 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 highest 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

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

Equality (1) follows from $N(\perp_u) = N(u)$ and the definition that $\perp_u$ does not label 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 $\rho$.
   Output: Optimal labeling $\mathcal{L}(\rho)$ of $T$.
1  $T' \leftarrow$ compute subdivision tree of $T$
2  for each leaf $v$ of $T'$ do $\mathcal{L}(v) \leftarrow \emptyset$
3  for each vertex $u$ of $T'$ considered in a bottom-up traversal of $T'$ do
4      $\mathcal{L}(u) \leftarrow \mathcal{L}(\texttt{OptCandidate}(u))$
5  return $\mathcal{L}(\rho)$
```

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; as we assume that each vertex has constant degree, $O(n^2)$ 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', because 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 3 *For a road map with a tree as underlying road graph with constant maximum degree,* MaxLabeledRoads *can be solved in $O(n^5)$ time using $O(n^2)$ space.*

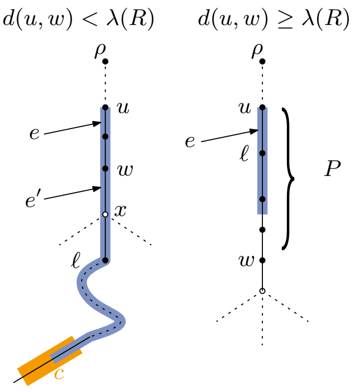
In case that all roads are paths, Algorithm 1 runs in $O(n^4)$ time, because for each $u \in V'$ the set $C(u)$ contains $O(n)$ labels. We shortly note that besides the *primary objective* to label a maximum number of road sections, Chirié [1] also suggested several additional *secondary objectives*, 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. In particular, as the primary and secondary objectives might contradict each other, the weight function can be used to express a compromise between them. The improvements that we present in the following sections are specialized for the objective of maximizing the number labeled road sections.

4.2 Improvements on Running Time

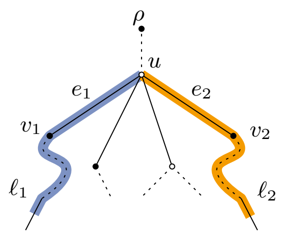
Next we describe how the running time of Algorithm 1 can be improved to $O(n^3)$ time by speeding up OptCandidate(u) to $O(n)$ time.

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

(i) $t(\ell) = \mathsf{E}(u)$,
(ii) $h(\ell)$ is located on a road section but not on a junction edge, and

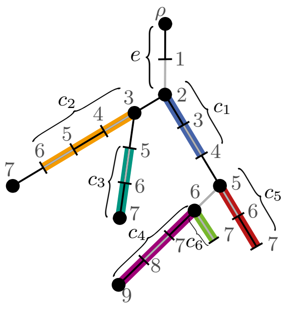


(a) Case 1: Regular vertex.

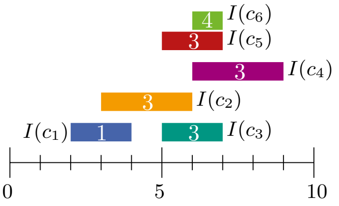


(b) Case 2: Junction vertex.

Fig. 9 Application of linearizations. **a** The vertex u is a regular vertex. Hence, only vertical labels can end at u. **b** The vertex u is a junction vertex. Hence, only horizontal labels can cover u



(a) Tree representation.



(b) Interval representation.

Fig. 10 Schematic illustration of e-docks. For simplicity the tree consists only of road sections (segments between black dots). The curves $c_1, \ldots, c_6$ are examples for e-docks. **a** The tree is annotated with distance marks. E.g., c_1 and c_2 superpose each other, while c_1 and c_5 do not. **b** Each e-dock is annotated with its weight. For example the e-dock c_6 is maximal, as for every point p of c_6 there is a maximal e-rooted curve ending at p. Two e-docks superpose each other if their distance intervals intersect

(iii) $\text{length}(\mathsf{E}(e) \cap \ell) = \min\{\text{length}(\ell), \text{length}(\mathsf{E}(e))\}$, i.e., ℓ emanates from $\mathsf{E}(u)$ passing through e.

For instance, the red label in Fig. 8b is an e-rooted curve. 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 highest point of ℓ; see Fig. 9a, b, respectively. Moreover, an e-rooted curve ℓ is *maximal* if there is no other e-rooted curve ℓ' with $\text{length}(\ell) = \text{length}(\ell')$ and $\omega(\mathcal{L}(\ell')) > \omega(\mathcal{L}(\ell))$.

For a sub-curve $c \subseteq \mathsf{E}(e')$ of a road section e' we say that it is an *e-dock* if for any two e-rooted curves ℓ and ℓ' that end on c we have $\omega(\mathcal{L}(\ell)) = \omega(\mathcal{L}(\ell'))$; we call

$\omega(\mathcal{L}(\ell))$ the *weight* $\omega_e(c)$ of c. See Fig. 10 for an illustration. Further, an e-dock c is *maximal* if for every point p on c there is a maximal e-rooted curve ℓ that ends on c, i.e., $h(\ell) = p$. Hence, with a maximal e-dock c in $\mathsf{E}(T)$ we can represent all maximal e-rooted curves that end on c without storing them explicitly.

Moreover, we are only interested in maximal e-docks such that no two maximal e-docks represent maximal e-rooted curves of the same length. To that end, we define the *distance interval I(c)* of an e-dock as $[\,\mathrm{d}_{t(c)}, \mathrm{d}_{h(c)}]$. Since T is a tree, for every point p of c we have $\mathrm{d}_p \in I(c)$. We say that two e-docks c and c' *superpose* each other if $I(c) \cap I(c') \neq \emptyset$. Hence, two maximal e-docks that do not superpose each other represent maximal e-rooted curves of different lengths.

Next, we introduce a data structure that encodes for each edge $e = (u, v)$ of T all maximal e-rooted curves as $O(n)$ pairwise superposition-free maximal e-docks in $\mathsf{E}(T_u)$.

Definition 1 (*Linearization*) Let $e = (u, v)$ be an edge of T. A tuple $(L, \overline{\omega} : L \rightarrow \mathbb{N})$ is called a *linearization* of e if L is a set of pairwise superposition-free maximal e-docks such that

(1) For each e-dock $c \in L$ it holds that $\overline{\omega}(c) = \omega_e(c)$, and
(2) For any e-rooted curve ℓ there is an e-dock $c \in L$ with $\mathrm{length}(\ell) + \mathrm{d}_u \in I(c)$.

The main idea of the linearization is that for each length of a possible e-rooted curve there is an e-dock in L that represents the best possible weight of an e-rooted curve with the same length. To that end, Condition (1) enforces that the function $\overline{\omega}$ actually encodes the weights of the e-docks in L. Condition (2) ensures that L is complete in the sense that for any e-rooted curve ℓ there is an e-rooted curve ℓ' of the same length that ends on an e-dock of L. We observe that ℓ' is unique as L contains pairwise superposition-free e-docks.

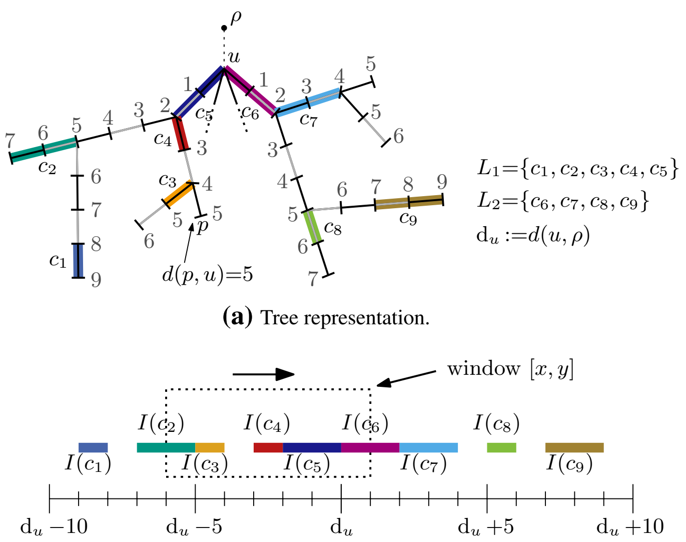
Assume that we apply Algorithm 1 on T' and that we currently consider the vertex u of T'. Hence, we can assume that for each vertex $v \neq u$ of T'_u its optimal candidate and $\omega(\mathcal{L}(v))$ is given. We first explain how to speed up `OptCandidate` using linearizations. Afterwards, we present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

4.2.1 Speeding up `OptCandidate` with Linearizations

Here we assume that the linearizations are given for the edges of T. Depending on 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 is 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. 9a.

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.



ρ
u
$L_1 = \{c_1, c_2, c_3, c_4, c_5\}$
$L_2 = \{c_6, c_7, c_8, c_9\}$
$d_u := d(u, \rho)$
$d(p, u) = 5$
window $[x, y]$
$I(c_1)$ $I(c_2)$ $I(c_3)$ $I(c_4)$ $I(c_5)$ $I(c_6)$ $I(c_7)$ $I(c_8)$ $I(c_9)$
$d_u - 10$ $d_u - 5$ d_u $d_u + 5$ $d_u + 10$

(a) Tree representation.

(b) Interval representation.

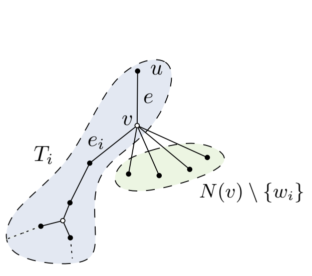
Fig. 11 Constructing the optimal candidate of u based on the linearizations $(L_1, \overline{\omega}_1)$ and $(L_2, \overline{\omega}_2)$. The tree is annotated with distance marks

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 e-dock $c \in L$ 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 e-dock $c \in L$ its distance interval $I(c)$ and check whether there is a $t \in I(c)$ 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 $\mathsf{E}(T)$.

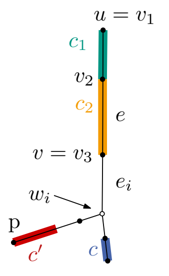
If such an e-dock c exists, by definition of a linearization the optimal candidate is either $\perp_u$ or ℓ, which we can decide in $O(1)$ time by checking whether $\omega(\mathcal{L}(\perp_u)) \geq \omega(\mathcal{L}(\ell))$. Note that we have $\omega(\mathcal{L}(\perp_u)) = \omega(\mathcal{L}(v))$ 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.

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. 9b. 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.

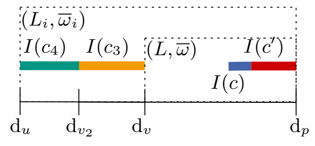
Let e_1 and e_2 be two outgoing edges of u that belong to the same road R, and let $(L_1, \overline{\omega}_1)$ and $(L_2, \overline{\omega}_2)$ be the linearizations of e_1 and e_2, respectively. We now define for e_1 and e_2 and their linearizations the operation opt-cand(L_1, L_2) that returns an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



(a) e is a junction edge.



(b) e is a road section.



(c) Interval representation of (b).

Fig. 12 First step of constructing a linearization: for each edge e_i its linearization $(L, \overline{\omega})$ is extended to a linearization $(L_i, \overline{\omega}_i)$ of the tree T_i rooted at u

For $i \in \{1, 2\}$ 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. 11. 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 $\lambda(R)$, if $\mathrm{d}_u \in [x, y]$, and if there is an e_1-dock $c_1 \in L_1$ and an e_2-dock $c_2 \in L_2$ with $x \in I(c_1)$ and $y \in I(c_2)$; see Fig. 11. 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 $\text{length}(\ell_1) + \text{length}(\ell_2) = \lambda(R)$. 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 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 opt-cand(L_1, L_2), which yields 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 we assume that every 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.*

4.2.2 Construction of Linearizations

We now show how to recursively construct a linearization for an edge $e = (u, v)$ of T. To that end we assume that we are given the subdivision tree T' of T and the linearizations for the outgoing edges $e_1 = (v, w_1), \ldots, e_k = (v, w_k)$ of v that belong to the same road R as e. Further, we can assume that we have computed the weight $\omega(\mathcal{L}(w))$ for each vertex w in T'_u except for u. In case that two of these vertices share the same position in $\mathsf{E}(T'_u)$ we remove the one with less weight. Let

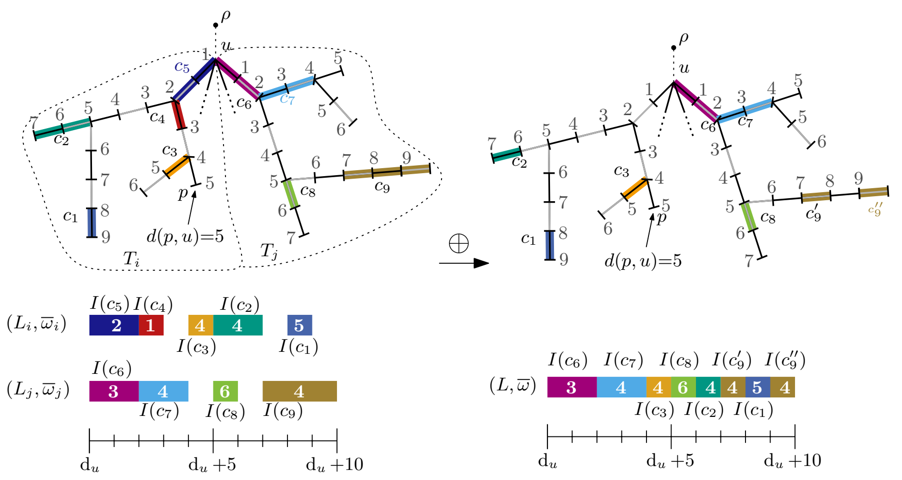


Fig. 13 Second step of constructing a linearization: Merging the linearizations of the trees T_i and T_j

T_i be the tree induced by the edges e, e_i and the edges of the subtree rooted at w_i. As a first step we compute for each linearization $(L, \overline{\omega})$ of each edge e_i a linearization $(L_i, \overline{\omega}_i)$ for e restricted to tree T_i, i.e., conceptually, we assume that T_u consists only of T_i's edges.

If e is a junction edge (see Fig. 12a), we set $L_i \leftarrow L$ and determine the weight of each curve $c \in L_i$ as

$$\overline{\omega}_i(c) \leftarrow \overline{\omega}(c) + \sum_{w \in N(v) \setminus \{w_i\}} \omega(\mathcal{L}(w)).$$

Otherwise, if e is a road section, let $v_1, \dots, v_l$ be the vertices of the subdivision tree T' that lie on e, i.e., $\mathsf{E}(v_j) \in \mathsf{E}(e)$ for all $1 \le j \le l$; see Fig. 12b–c. We assume that $\mathrm{d}(v_1) < \ldots < \mathrm{d}(v_l)$, which in particular yields $v_1 = u$ and $v_l = v$. Let c_1 be the curve $\mathsf{E}((v_1, v_2))$ and for $2 \le j < l$ let c_j be the curve $\mathsf{E}((v_j, v_{j+1})) \setminus \mathsf{E}(v_j)$. Hence, we have $\bigcup_{j=1}^{l} c_j = \mathsf{E}(e)$ and $c_j \cap c_{j'} = \emptyset$ for $1 \le j < j' < l$. We set

$$L_i \leftarrow L \cup \bigcup_{j=1}^{l-1} \{c_j\}.$$

We weight each curve $c \in L_i$ as follows. If c is contained in L, we set $\overline{\omega}_i(c) \leftarrow \overline{\omega}(c) + 1$.

Otherwise, c is a sub-curve of $\mathsf{E}(e)$ and there exists a j with $c = c_j$. We set $\overline{\omega}_i(c) \leftarrow \omega(\mathcal{L}(v_{j+1}) \cup \{\ell_c\})$, where $\ell_c \subseteq \mathsf{E}(e)$ is an e-rooted curve that starts at $\mathsf{E}(u)$ and ends on c. The next lemma shows that this transformation yields a linearization as desired.

Lemma 4 *For each outgoing edge e_i with linearization $(L, \overline{\omega})$, the tuple $(L_i, \overline{\omega}_i)$ is a linearization of e restricted to the tree T_i.*

Proof We use the same notation as used above and prove the following three statements:

(A) The curves in L_i are pairwise superposition-free.
(B) The curves in L_i are maximal e-docks with weight $\omega_e(c) = \overline{\omega}_i(c)$.
(C) For each e-rooted curve ℓ there is an e-dock $c \in L$ with length $(\ell) + \mathrm{d}_u \in I(c)$.

(A) The curves in L_i are pairwise superposition-free. Since L contains only curves that do not superpose each other, the only curves that could superpose another curve in L_i are contained in $L_i \backslash L$. Since $L_i \backslash L$ is empty for a junction edge, we can assume that e is a road section. By construction these curves in $L_i \backslash L$ partition $\mathsf{E}(e)$ without intersecting each other. Further, by assumption no two road sections share a common vertex and since all curves of L are contained in $\mathsf{E}(T_v)$, the curves in $L_i \backslash L$ cannot superpose any curve in L.

(B) The curves in L_i are maximal e-docks with $\omega_e(c) = \overline{\omega}(c)$. First of all, all curves in L_i are sub-curves of road sections: since L is a linearization, each curve of L must be a sub-curve of a road section. Further, if e is a road section, the curves $L_i \backslash L$ are sub-curves of e and otherwise, if e is a junction edge, the set $L_i \backslash L$ is empty. Next, we show that for each point p of any curve $c \in L_i$ there is a maximal e-rooted curve ℓ that ends at p with $\omega(\mathcal{L}(\ell)) = \overline{\omega}_i(c)$. This particularly shows that c is an e-dock as any two e-rooted curves ending on c have consequently the same weight. We distinguish the two cases that e is a road section or a junction edge.

Case: e is a road section. We distinguish the two cases that $c \in L_i \backslash L$ or $c \in L$. First, let c be an arbitrary curve in $L_i \backslash L$ and let ℓ be any e-rooted curve that ends on c. Obviously, ℓ must be a maximal e-rooted curve, because there is no other point in $\mathsf{E}(T_i)$ having the same distance to ρ as $h(\ell)$ has. We show that $\omega_e(c) = \overline{\omega}_i(c)$. Let $v_1, \dots, v_l$ be the vertices of the subdivision tree T' that lie on e as defined above. By construction there is an edge (v_j, v_{j+1}) with $1 \le j < l$ and $c \subseteq \mathsf{E}(v_j, v_{j+1})$. It holds that

$$\omega_e(c) = \omega(\mathcal{L}(\ell)) = \omega(\mathcal{L}(v_{j+1}) \cup \{\ell\}) = \overline{\omega}_i(c).$$

Hence, we obtain that c is an e-dock with weight $\omega_e(c) = \overline{\omega}_i(c)$.

Now, consider a curve $c \in L$ and let ℓ be any e-rooted curve that ends on c. As L is a linearization of e_i, for each point p on c there must be a maximal e_i-rooted curve ℓ' with $h(\ell') \in c$. We choose ℓ' such that $h(\ell') = h(\ell)$. Since ℓ' is a maximal e_i -rooted curve, the curve ℓ must be a maximal e-rooted curve. Further, ℓ labels one road section more than ℓ'. Hence, we obtain

$$\omega_e(c) = \omega(\mathcal{L}(\ell)) = \omega(\mathcal{L}(\ell')) + 1 = \overline{\omega}(c) + 1 = \overline{\omega}_i(c).$$

Hence, we again obtain that c is a maximal e-dock with weight $\omega_e(c) = \overline{\omega}_i(c)$.

Case: e is a junction edge. Let c be a curve in L_i and let ℓ be any e-rooted curve that ends on c. Further, let ℓ' be the e_i-rooted sub-curve of ℓ that starts at $\mathsf{E}(v)$ and ends at $h(\ell)$; by definition of L such a curve exists. It holds

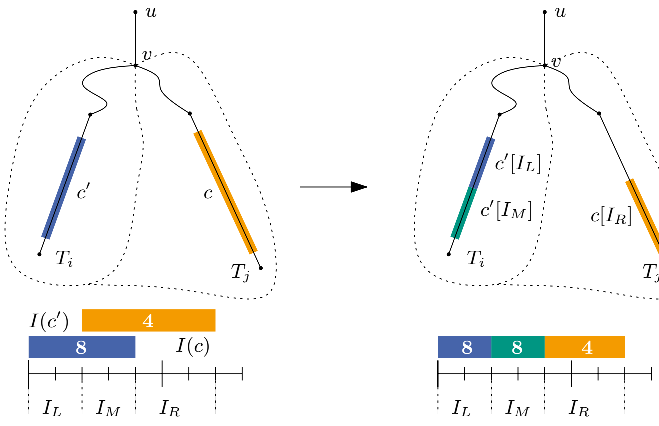


Fig. 14 Illustration of merging two linearizations $(L_i, \overline{\omega}_i)$ and $(L_j, \overline{\omega}_j)$ into one linearization $(L_1, \overline{\omega}_i)$. The trees are annotated with distance marks

$$\omega_e(c) = \omega(\mathcal{L}(\ell)) = \omega(\mathcal{L}(\ell')) + \sum_{w \in N(v) \setminus \{w_i\}} \omega(\mathcal{L}(w)) = \overline{\omega}(c)$$
$$+ \sum_{w \in N(v) \setminus \{w_i\}} \omega(\mathcal{L}(w)) = \overline{\omega}_i(c)$$

Since ℓ' is a maximal e_i-rooted curve, it directly follows that ℓ is a maximal e-rooted curve with respect to T_i. Hence, we obtain that c is an e-dock with weight $\omega_e(c) = \overline{\omega}_i(c)$.

(C) For each e-rooted curve ℓ there is an e-dock $c \in L$ with length $(\ell) + \mathrm{d}_u \in I(c)$. First consider an e-rooted curve ℓ that either ends on e_i or on an edge of T_{w_i}. Recall that $h(\ell)$ must lie on a road section. Then there is an e_i-rooted curve ℓ' with $\ell' \subseteq \ell$ and $h(\ell) = h(\ell')$. Hence, there is a curve $c \in L$ with length $(\ell') + \mathrm{d}_v \in I(c)$. Since ℓ' is a sub-curve of ℓ, we also have length $(\ell) + \mathrm{d}_u \in I(c)$. Now, consider an e-rooted curve ℓ that ends on e. In that case e is a road section. By construction there is a curve $c \in L_i \setminus L$ with length $(\ell) + \mathrm{d}_u \in I(c)$. □

In the next step we define an operation $\oplus$ by means of which two linearizations $(L_i, \overline{\omega}_i)$ and $(L_j, \overline{\omega}_j)$ can be combined to one linearization $(L_i, \overline{\omega}_i) \oplus (L_j, \overline{\omega}_j)$ of e that is restricted to the subtree $T_{i,j}$ induced by the edges of T_i and T_j. Consequently, $\bigoplus_{i=1}^{k} (L_i, \overline{\omega}_i)$ is the linearization of e without any restrictions.

We define $(L, \overline{\omega}) = (L_i, \overline{\omega}_i) \oplus (L_j, \overline{\omega}_j)$ as follows; for an illustration see also Figs. 13 and 14. Let $c_1, \dots, c_\ell$ be the curves of $L_i \cup L_j$ such that for any two curves c_s, c_t with $s < t$ the left endpoint of $I(c_s)$ lies to the left of the left endpoint of $I(c_t)$; ties are broken arbitrarily. We successively add the curves to L in the given order enforcing that the curves in L remain superposition-free. Let c be the next curve to be added to L.

Without loss of generality, let $c \in L_i$. The opposite case can be handled analogously. In case that there is no curve superposing c, we add c to L and set $\overline{\omega}(c) = \overline{\omega}_i(c)$. If c superposes a curve in L, due the order of insertion, there can only be one curve c' in L that superposes c. First we remove c' from L. Let I_M be the interval describing the set $I(c) \cap I(c')$, and let I_L and I_R be the intervals describing the set $I(c) \cup I(c') \backslash (I(c) \cap I(c'))$ such that I_L lies to the left of I_M and I_R lies to the right of I_M.

We now define three curves c_L, c_M and c_R with $I(c_L) = I_L$, $I(c_M) = I_M$ and $I(c_R) = I_R$ such that each of these three curves is a sub-curve of either c or c'. To that end let $c[I]$ denote the sub-curve of c whose distance interval is I. We define the curve c_R with weight $\overline{\omega}(c_R)$ as

$$(c_R, \overline{\omega}(c_R)) = \begin{cases} (c[I_R], \overline{\omega}_i(c)), & \text{if } I_R \subseteq I(c) \\ (c'[I_R], \overline{\omega}(c')), & \text{if } I_R \subseteq I(c'). \end{cases}$$

The curve c_L and its weight $\overline{\omega}(c_L)$ is defined analogously. The curve c_M and its weight $\overline{\omega}(c_M)$ is

$$(c_M, \overline{\omega}(c_M)) = \begin{cases} (c[I_M], \overline{\omega}_i(c)), & \text{if } \overline{\omega}_i(c) \geq \overline{\omega}(c') \\ (c'[I_M], \overline{\omega}(c')), & \text{if } \overline{\omega}_i(c) < \overline{\omega}(c'). \end{cases}$$

The next lemma proves that $(L_i, \overline{\omega}_i) \oplus (L_j, \overline{\omega}_j)$ is a restricted linearization.

Lemma 5 *Let $(L_i, \overline{\omega}_i)$ and $(L_j, \overline{\omega}_j)$ be two linearizations of $e = (u, v)$ that are restricted to the trees T_i and T_j, respectively. Then $(L, \overline{\omega}) = (L_i, \overline{\omega}_i) \oplus (L_j, \overline{\omega}_j)$ is a linearization of e restricted to $T_{i,j}$. The operation needs $O(|L_i| + |L_j|)$ time.*

Proof First of all, the set L contains only curves that are pairwise free from any superpositions. This directly follows from the construction that curves c and c' superposing each other are replaced by three superposition-free curves c_L, c_M and c_R. Due to $I(c_L) \cup I(c_M) \cup I(c_R) = I(c) \cup I(c')$ the second condition of a linearization is satisfied. Further, as all curves in L_i and L_j are e-docks and as they remain the same or are shortened when added to L, all curves in L are also e-docks.

We finally prove that all e-docks are maximal and Condition (1) of a linearization is satisfied by doing an induction over the curves inserted to L. Let L^k be L after the k-th insertion step. Since L^0 is empty, the condition obviously holds for L^0. So assume that we insert c to L^k obtaining the set L^{k+1}. Without loss of generality assume that $c \in L_i$. If c does not superpose any curve in L^k, the condition directly follows from the definition of c. So assume that $c' \in L^k$ superposes c. Since $c \in L_i$, the curve c' is contained in $\mathsf{E}(T_j)$. We remove c' from L^k and insert the curves c_R, c_M and c_L as defined above. We prove that all three curves satisfy Condition (1).

Consider in the following the subtree $T_{i,j}$ of T_u restricted to the edges of T_i and T_j. We set $c_R = c[I_R]$ and set $\overline{\omega}(c_R) = \overline{\omega}_i(c)$, if $I_R \subseteq I(c)$. In that case there is no e-rooted curve $\ell \subseteq \mathsf{E}(T_j)$ with $\text{length}(\ell) + \mathrm{d}_u \in I_R$, i.e., either there is no curve ℓ in $\mathsf{E}(T_j)$ with $t(\ell) = \mathsf{E}(u)$ and $\text{length}(\ell) + \mathrm{d}_u \in I_R$, or any curve in $\mathsf{E}(T_j)$ with $t(\ell) = \mathsf{E}(u)$ and $\text{length}(\ell) + \mathrm{d}_u \in I_R$ ends on a junction edge. Consequently, any e-rooted curve ℓ with $\text{length}(\ell) + \mathrm{d}_u \in I_R$ and in particular any maximal e-rooted curve ℓ with $\text{length}(\ell) + \mathrm{d}_u \in I_R$ lies in $\mathsf{E}(T_i)$. Thus, the curve c_R is a maximal e-dock and satisfies Condition (1). For the case $I_R \subseteq I(c')$ and the curve c_L we can argue analogously.

So consider the curve c_M. Without loss of generality we assume that $\overline{\omega}_i(c) \geq \overline{\omega}(c')$. The opposite case can be handled analogously. For any maximal e-rooted curve ℓ in $\mathsf{E}(T_j)$ with length $(\ell) + \mathrm{d}_u \in I_M$ it must be true that $\omega(\mathcal{L}(\ell)) \leq \overline{\omega}(c_M)$. Further, since $c_M \subseteq c$ and c satisfies Condition (1) with respect to T_i, c_M is a maximal e-port and satisfies the Condition (1) with respect to $T_{i,j}$. □

Lemmas 4 and 5 yield that $\bigoplus_{i=1}^{k}(L_i, \omega_i)$ is the linearization of e without any restrictions. Computing it needs $O(\sum_{i=1}^{k} |L_i|)$ time.

Note that when computing optimal candidates (see Sect. 4.2.1) we are only interested in e-rooted curves ℓ that have length at most $\lambda(R)$, where R is the road of e. Hence, when constructing $(L_i, \overline{\omega}_i)$ for an edge e_i in the first step, we discard any curve c of L_i that does not allow an e-rooted curve that both ends on c and has length at most $\lambda(R)$; the curve c is not necessary for our purposes. Hence, we conceptually restrict T_i to the edges that are reachable from u by one label length. It is not hard to see that T' restricted to $\mathsf{E}(T_i)$ contains only $O(n)$ vertices, because each vertex of $V' \setminus V$ is induced by a chain of tightly packed vertical labels, whereas each label has length $\lambda(R)$. Hence, T' restricted to $\mathsf{E}(T_i)$ contains for each such chain at most one vertex of $V' \setminus V$. Further, the endpoints of the curves in L_i are induced by the vertices of T'. Hence, by discarding the unnecessary curves of L_i the set L_i has size $O(n)$. Altogether, by Lemma 5 and due to the constant degree of each vertex we can construct $\bigoplus_{i=1}^{k}(L_i, \omega_i)$ in $O(\sum_{i=1}^{k} n) = O(n)$ time.

When constructing $\mathcal{L}(u)$ for u as described in Algorithm 1, we first build the linearization L_e of each of u's outgoing edges. By Lemma 3 we can find in $O(n)$ time the optimal candidate of u. Then, due to the previous reasoning, the linearization of an edge of T and the optimal candidate of a vertex u can be constructed in $O(n)$ time. Altogether we obtain the following result.

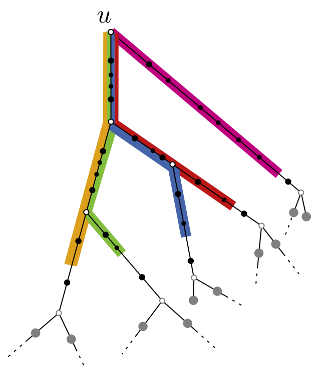
Proposition 1 *For a road map $\mathcal{M}$ with a tree T as underlying road graph,* MaxLabeledRoads *can be solved in $O(n^3)$ time.*

4.3 Improvements on Storage Consumption

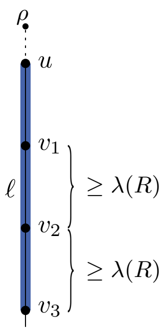
Since T' contains $O(n^2)$ vertices, the algorithm needs $O(n^2)$ space. This can be improved to $O(n)$ space. To that end T' is constructed *on the fly* while executing Algorithm 1. Parts of T' that become unnecessary are discarded. We prove that it is sufficient to store $O(n)$ vertices of T' at any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

When constructing the optimal labeling of T, we build for each edge (u, v) of T its linearization based on the linearization of the outgoing edges of v. Afterwards we discard the linearizations of these outgoing edges. Since we assume that each vertex has constant degree, it is sufficient to maintain a constant number of linearizations at any time if we consider the vertices of T' in an appropriate order.

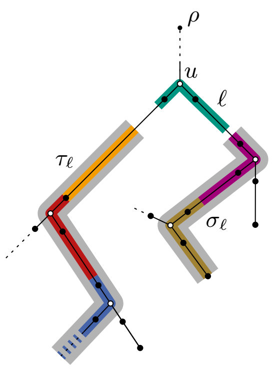
Hence, because each linearization has size $O(n)$, we need $O(n)$ space for storing the required linearizations in total. However, we store for each vertex u of T' the



(a) Reachable vertices.



(b) Lemma 6.



(c) Chains of label ℓ.

Fig. 15 Improvements on storage consumption. **a** Illustration of the reachable vertices from u. Vertices not reachable from u are marked gray. **b** Illustration of proof for Lemma 6. **c** Illustration for reconstructing the computed labeling (Color figure online)

weight $\omega(\mathcal{L}(u))$ and its optimal candidate. As T' has size $O(n^2)$ the space consumption is $O(n^2)$. In the following, we improve that bound to $O(n)$ space.

We call a vertex $v \in V'$ *reachable* from a vertex $u \in V'$ if there is a curve $\ell \subseteq \mathsf{E}(T'_u)$ that starts at $\mathsf{E}(u)$ and that is contained in the embedding of a road R with $\lambda(R) \geq \text{length}(\ell)$ such that $\mathsf{E}(v) \in \ell$ or $v \in N(\ell)$; see Fig. 15a. The set R_u contains all vertices of T'_u that are reachable from u. The next lemma shows that R_u has linear size.

Lemma 6 *For any vertex u of T' the set R_u has size $O(n)$.*

Proof Recall how T' is constructed: For each vertex $v \in V$ we construct a chain C of tightly packed vertical valid 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 labeled road sections.

Each label of such a chain C induces one vertex of T'. Hence, C induces a set V_C of vertices in T'. We show that for each chain C the set $V_C \cap \mathsf{R}_u$ contains at most two vertices. As we construct n chains in order to build T' the claim follows.

For the sake of contradiction assume that there is a chain C and a vertex u in T' such that $V_C \cap \mathsf{R}_u$ contains more than two vertices. Without loss of generality we assume that $V_C \cap \mathsf{R}_u$ contains three vertices, which we denote by v_1, v_2 and v_3; see Fig. 15b. We further assume that $\mathsf{d}_{v_1} < \mathsf{d}_{v_2} < \mathsf{d}_{v_3}$. By construction all labels in C lie in the embedding of the same road R_C, and $\mathsf{d}(v_1, v_2) \geq \lambda(R_C)$ and $\mathsf{d}(v_2, v_3) \geq \lambda(R_C)$. By definition of C there is a vertical curve $\ell \in \mathsf{E}(T'_u)$ that starts at $\mathsf{E}(u)$ and contains v_1, v_2 and v_3. Let e be the outgoing edge of u in T' whose embedding is covered by ℓ and consider the sub-curve $\ell' \subseteq \ell$ with length $\lambda(R_C)$ that starts at u. By definition of R_u, we know for each v_i with $1 \leq i \leq 3$ that either its embedding is contained in ℓ' or $v_i \in N(\ell')$. The definition of $N(\ell')$ implies the vertices of $N(\ell')$ cannot lie on a common vertical curve in T. Hence, as v_1, v_2 and v_3 lie on the vertical curve

ℓ, only v_3 can be contained in $N(\ell')$. Hence, $\mathsf{E}(v_1), \mathsf{E}(v_2) \in \ell'$. Further, because $v_2 \notin N(\ell')$, we have $\mathsf{E}(v_2) \neq h(\ell')$, which implies $\mathrm{d}(v_1, v_2) < \lambda(R)$ and contradicts $\mathrm{d}(v_1, v_2) \geq \lambda(R)$. □

Assume that we apply Algorithm 1 considering vertex u. When constructing u's optimal candidate, by Lemma 6 it is sufficient to consider the vertices of T'_u that lie in R_u. On that account we discard all vertices of T'_u that lie in $V' \backslash V$, but not in R_u. Further, we compute the vertices of $V' \backslash V$ that subdivide the incoming edge $(t, u) \in E$ *on demand*, i.e., we compute them when constructing the optimal candidate of t. Hence, the space consumption is linear.

However, when discarding vertices of T', we lose the possibility of reconstructing the labeling. We therefore annotate each vertex $u \in V$ of the original tree T with further information. To that end consider a canonical labeling $\mathcal{L}$ of T. Let ℓ be a horizontal label of $\mathcal{L}$ and let e be the edge of T on which ℓ's head is located; see Fig. 15c. Either, no other label of $\mathcal{L}$ ends on e, or another label ℓ' ends on e that belongs to a chain σ_ℓ of tightly packed vertical labels. Analogously, we can define the chain τ_ℓ with respect to edge e' on which $\ell's$ tail is located. On that account we store for a junction vertex $u \in V$ not only its optimal candidate $\ell \in C(u)$, but also the two chains σ_ℓ and τ_ℓ, if they exist. Note that such a chain of tightly packed vertical labels is uniquely defined by its start and endpoint, which implies that $O(1)$ space is sufficient to store both chains. Using a breadth-first search we can easily reconstruct these chains in linear time. For a regular vertex $u \in V$ we analogously store the chain σ_ℓ of its optimal candidate $\ell \in C(u)$ if it exists. Since ℓ is vertical, we do not need to consider its tail. For the special case that $\ell = \bot_u$, we define that σ_ℓ is the chain of tightly packed vertical labels that ends on the only outgoing edge e of u. Summarizing, the additional information together with the optimal candidates stored at the vertices of the original tree are sufficient to reconstruct the labeling of T. Together with Proposition 1 we obtain the following result.

Theorem 4 *For a road map $\mathcal{M}$ with a tree T as underlying road graph with constant maximum degree,* MaxLabeledRoads *can be solved in $O(n^3)$ time using O(n) space.*

5 Conclusions

In this paper, we investigated the problem of maximizing the number of labeled road sections in a labeling of a road map. We showed that this problem is NP-hard in general. For the special case of trees we introduced a dynamic programming algorithm. This algorithm utilizes the basic observation that a label passing through a vertex splits the tree into sub-instances that can be considered independently. By systematically exploring all $O(n^2)$ possibilities how a label can pass through a vertex, the dynamic programming approach recursively computes an optimal solution in $O(n^5)$ time. We improved that bound to $O(n^3)$ by using the new concept of linearizations. Finally, we showed that by computing the necessary data structures *on the fly*, the approach needs linear space while maintaining $O(n^3)$ running time.

The underlying road graphs of real-world road maps are rarely trees. However, in an applied companion paper [6] we show that road maps decompose into a large number of subgraphs by placing trivially optimal road labels and removing the corresponding edges from the graph. It turns out that a vast majority of the resulting subgraphs are actually trees, which we can label optimally by the algorithm proposed here. As a consequence, this means that a large fraction of all road sections in our real-world road graphs can be labeled optimally by combining this simple preprocessing strategy with our tree labeling algorithm. For the remaining subgraphs we observe that a majority are nearly trees in the sense that the cycles can be broken by removing only few edges. Hence, a future research direction points to the investigation of MaxLabeledRoads for such tree-like graphs. In addition, the investigation of MaxLabeledRoads for cactus graphs and graphs with bounded tree-width promises new interesting insights. Is MaxLabeledRoads already NP-hard for these graphs, or can our dynamic programming approach be adapted accordingly?

In the presented model, we maximize the number of labeled road sections, which is a rather simple optimization function. From a cartographic point of view more complex cost functions (e.g., priority of road sections, shape of labels, etc.) may also be of interest. Hence, the question arises whether our results carry over. The NP-hardness result strongly relies on the assumption that the number of labeled road sections is maximized. As a consequence, the computational complexity of the problem cannot be directly transferred, but must be considered for each cost function individually. Still, we believe that the basic constructions of the presented gadgets may help to find NP-hardness proofs for newly considered cost functions.

In contrast to the NP-hardness proof, the basic dynamic programming approach for trees is more likely to carry over. As long as a cost-optimal labeling can be composed by a discrete and pre-defined set of label candidates, the dynamic programming idea can be used to compute it. However, the running time strongly relies on the number of label candidates to be considered. Hence, minimizing the number of label candidates a priori becomes an important research question. In particular, the question arises whether the introduced speed-up techniques (e.g., the use of linearizations) carry over.

In the last decades maps have undergone a dramatic change from static figures to dynamic visualizations in which the scale, rotation and view of the map changes over time. Accordingly, in order to obtain a temporally coherent label placement, the change of the map needs to be taken into account. While dynamic maps have been extensively investigated for labeling point features, labeling line features has hardly been considered. Hence, an interesting and important research direction is to adapt the presented model and algorithms to dynamic maps.

Acknowledgements Open Access funding provided by Projekt DEAL.

References

1. Chirié, F.: Automated name placement with high cartographic quality: city street maps. Cartogr. Geogr. Inf. Sci. **27**(2), 101–110 (2000)
2. Edmondson, S., Christensen, J., Marks, J., Shieber, S.M.: A general cartographic labelling algorithm. Cartographica **33**(4), 13–24 (1996)
3. Imhof, E.: Positioning names on maps. Am. Cartogr. **2**(2), 128–144 (1975)
4. Lichtenstein, D.: Planar formulae and their uses. SIAM J. Comput. **11**(2), 329–343 (1982)
5. Maass, S., Döllner, J.: Embedded labels for line features in interactive 3d virtual environments. In: Computer Graphics, Virtual Reality, Visualisation and Interaction (AFRIGRAPH'07), pp. 53–59. ACM Press (2007)
6. Niedermann, B., Nöllenburg, M.: An algorithmic framework for labeling road maps. In: Geographic Information Science (GIScience'16), Volume 9927 of Lecture Notes in Computer Science, pp. 308–322. Springer (2016)
7. Neyer, G., Wagner, F.: Labeling downtown. In: Algorithms and Complexity (CIAC'00), Volume 1767 of Lecture Notes in Computer Science, pp. 113–124. Springer, Berlin (2000)
8. Reimer, A., Rylov, M.: Point-feature lettering of high cartographic quality: a multi-criteria model with practical implementation. In: European Workshop on Computational Geometry (EuroCG'14) (2014)
9. Schwartges, N., Morgan, B., Haunert, J.-H., Wolff, A.: Labeling streets along a route in interactive 3D maps using billboards. In: AGILE 2015, Lecture Notes in Geoinformation and Cartography, pp. 269–287. Springer (2015)
10. Strijk, T.: Geometric algorithms for cartographic label placement. Dissertation, Utrecht University (2001)
11. Seibert, S., Unger, W.: The hardness of placing street names in a Manhattan type map. Theor. Comput. Sci. **285**, 89–99 (2002)
12. Schwartges, N., Wolff, A., Haunert, J.-H.: Labeling streets in interactive maps using embedded labels. In: Advances in Geographic Information Systems (ACM-GIS'14), pp. 517–520. ACM Press (2014)
13. van Kreveld, M.: Geographic information systems. In: Goodman, J.E., O'Rourke, J., Tóth, C.D. (eds.) Handbook of Discrete and Computational Geometry. Chapter 58, 2nd edn, pp. 1293–1314. Chapman and Hall/CRC, Boca Raton, FL (2010)
14. Vaaraniemi, M., Treib, M., Westermann, R.: Temporally coherent realtime labeling of dynamic scenes. In: Computing Geospatial Research Applications (COM.Geo'12), pp. 17:1–17:10. ACM Press (2012)
15. Wolff, A., Knipping, L., van Kreveld, M., Strijk, T., Agarwal, P.K.: A simple and efficient algorithm for high-quality line labeling. In: Innovations in GIS VII: GeoComputation. Chapter 11, pp. 147–159. Taylor & Francis (2000)
16. Wolff, A., Strijk, T.: The map labeling bibliography. http://liinwww.ira.uka.de/bibliography/Theory/map.labeling.html (2009). Accessed 27 Jan 2020