DOI: 10.1111/cgf.12921
Eurographics Conference on Visualization (EuroVis) 2016
K.-L. Ma, G. Santucci, and J. van Wijk
(Guest Editors)

Volume 35 (2016), Number 3

Location-dependent generalization of road networks based on equivalent destinations

Thomas C. van Dijk[1], Jan-Henrik Haunert[2] and Johannes Oehrlein[2]

[1]Chair for Computer Science I, University of Würzburg, Germany
thomas.van.dijk@uni-wuerzburg.de
[2]Institute for Geoinformatics and Remote Sensing, University of Osnabrück, Germany
{janhhaunert,johannes.oehrlein}@uni-osnabrueck.de

Abstract

Suppose a user located at a certain vertex in a road network wants to plan a route using a wayfinding map. The user's exact destination may be irrelevant for planning most of the route, because many destinations will be equivalent in the sense that they allow the user to choose almost the same paths. We propose a method to find such groups of destinations automatically and to contract the resulting clusters in a detailed map to achieve a simplified visualization. We model the problem as a clustering problem in rooted, edge-weighted trees. Two vertices are allowed to be in the same cluster if and only if they share at least a given fraction of their path to the root. We analyze some properties of these clusterings and give a linear-time algorithm to compute the minimum-cardinality clustering. This algorithm may have various other applications in network visualization and graph drawing, but in this paper we apply it specifically to focus-and-context map generalization. When contracting shortest-path trees in a geographic network, the computed clustering additionally provides a constant-factor bound on the detour that results from routing using the generalized network instead of the full network. This is a desirable property for wayfinding maps.

Keywords: clustering, simplification, map generalization

Categories and Subject Descriptors (according to ACM CCS): I.2.1 [ARTIFICIAL INTELLIGENCE]: Applications and Expert Systems—Cartography, I.5.3 [PATTERN RECOGNITION]: Clustering—Algorithms

1. Introduction

Suppose a user wants to get by car from Berlin, Germany to a certain address in Groningen, The Netherlands. She would like to plan her route using a wayfinding map. However, she does not want to be bothered with a detailed map of all of Europe. At the start of her journey, the exact address in Groningen is irrelevant to her, since it would not influence the direction she should start driving in – or, in fact, even where to drive within, say, the next hour. Therefore, the map in a dynamic visualization system may well collapse the city (or even the province) Groningen to a single vertex: any particular destination within Groningen leads to the exact same wayfinding decisions at this point. Only later does the exact address in Groningen become relevant: the map representation of Groningen should change while the user approaches her destination.

In this paper we formally define whether it makes sense to distinguish between a set of destinations. Let the graph $G = (V, E)$ represent the road network and $s \in V$ be the user's current location. Our idea is to compare, for any two vertices $u, v \in V$, the shortest path P_{su} from the user's position s to u, and the shortest path P_{sv} from s to v. If P_{su} *shares* a user-specified fraction $\alpha \in [0, 1]$ of its length with P_{sv} and vice versa, we consider u and v to be equivalent destinations, whose distinction from the user's current perspective is not meaningful. This definition makes sense if α is set to a relatively high number, for example if $\alpha = 0.95$. With this setting, the user can safely navigate for a relatively long distance without having to bother about the distinction between u and v. We consider a set $S \subseteq V$ of destinations equivalent (with respect to a start vertex s and a threshold α) if all vertices in S are mutually equivalent. We will reduce the level of detail of a network by using this equivalence condition to decide which locations can be aggregated.

The reduction of the level of detail of a map is a classical problem in cartography, where it is commonly termed *map generalization* and often approached by optimization [WJT03, Ses05]. Because map generalization has turned out to be a highly complex problem, however, it is typically subdivided into multiple tasks such as the selection of objects for the output map [TP66], the aggregation of objects [HW10], and the simplification of lines [DDS09]. When generalizing geographic networks, the first step is usually to select edges of a graph representation of the network. This step aims to reduce the visual clutter in the output map while preserving characteristic properties and high-level structures of the

network, for example, connectivity [MB93, Zha05] and sequences of line segments that are perceived as groups [TB02]. Often an aim is to preserve vertices or edges of high importance, which can be defined based on graph theoretical centrality measures [JC04]. Brunel et al. [BGK*14] modeled the generalization of networks as optimization problems, showed $\mathcal{NP}$-hardness for several of their models, and developed approximation algorithms as well as efficient heuristics. Chimani et al. [CvDH14] considered a problem in which the edges of a graph have to be removed iteratively to produce a sequence of subsequently more generalized maps of a network. They also showed $\mathcal{NP}$-hardness for their model and developed efficient approximation algorithms and heuristics.

In this paper, we develop a new model and generalization algorithms based on optimization for focus-and-context network maps, in which some parts of a network are represented with more details than other parts. Such maps have often been suggested for navigation and wayfinding tasks [ZR02], for example, to allow a user to follow a given route [AS01]. Kopf et al. [KAB*10] have presented a method for the automatic generation of *destination maps*, which allow different users to find routes to a given destination. For this purpose, the road network is presented with more details in the destination's vicinity. Focus-and-context maps are often distorted to present highly detailed parts of a network map at a larger scale than other parts. However, we consider the generalization of the network as an interesting problem of its own. After the map has been generalized with our method, different distortion techniques could be applied to use the available space in an optimal way and to reduce the remaining visual clutter, for example, a fish-eye projection [YOT09] or an optimization approach [vDH14]. An overview of distortion techniques for focus-and-context visualization is provided by Cockburn et al. [CKB09].

To summarize, in existing generalization systems, the level of detail is often dictated by a selected map scale. In this paper, however, we develop a method that selects details based on whether they are informative with respect to wayfinding tasks. Unnecessary details are removed and, thus, the visual complexity of the map is greatly reduced. We do not explicitly model graphical conflicts, however, since those may be resolved by selecting an appropriate map scale or using existing distortion techniques.

Our method detects clusters that exist in a road network and in this sense is related to speed-up techniques for shortest-path computations based on contraction hierarchies [GSSV12] or highway hierarchies [SS12]. Such concepts, however, are not readily applicable for the generation of focus-and-context maps of networks.

While this paper primarily considers the generalization of road networks, many other applications also require the visualization of hierarchical information and as such this topic has attracted much attention. Examples include the drawing of business data [VvWvdL06], phylogenetic trees [HRR*07] and file systems [BYB*13]. Particularly the latter two applications typically involve large tree structures where the use of focus-and-context techniques is appropriate if not outright necessary. The concept of focus and context in tree visualization can be traced back at least to Furnas, who proposed fish-eye views where nodes have an importance based on their distance to a focus node [Fur86]. This can be achieved, for example, using hyperbolic trees [LRP95]. These approaches do not explicitly generalize the hierarchical structure, but instead rely on scaling all available node-link information. An alternative drawing style consists of variations of so-called treemaps, for which focused visualizations have also been investigated [BL07].

We now come to the results presented in this paper. We are given a graph $G = (V,E)$ representing the road network, a source vertex $s \in V$, a number $\alpha \in [0,1]$, and weights $w\colon E \to \mathbb{R}_{\geq 0}$ reflecting edge lengths. In a geographic network, these weights can for example be Euclidean distances, possibly weighted by road class or travel time. Our task is then to partition the vertex set V automatically such that each cell corresponds to a set of equivalent destinations. (Singleton cells are allowed.) Subject to this constraint, we will minimize the number of cells in the partition: this generalizes the road network as much as possible. Since the equivalence condition is based on sharing a *fraction* of a shortest path, clusters near the source s must likely be small and many. Farther away, an optimal solution to the problem will probably have clusters that reflect larger geographic structures.

First we formally define this clustering problem in its general setting on arbitrary trees (Section 2) and give an efficient linear-time algorithm that solves this problem (Section 3). We also observe that a structural theorem plus known algorithms would give another polynomial-time algorithm, but that this is more complicated and less efficient. Then we apply this clustering algorithm specifically to modified shortest-path trees in geographic road networks (Section 4). We give several options for visualizing the resulting generalized tree and show how to incorporate cross-tree connectivity information in the drawing. We close with concluding remarks and an outlook on directions for future work (Section 5).

2. Equivalent destinations in trees

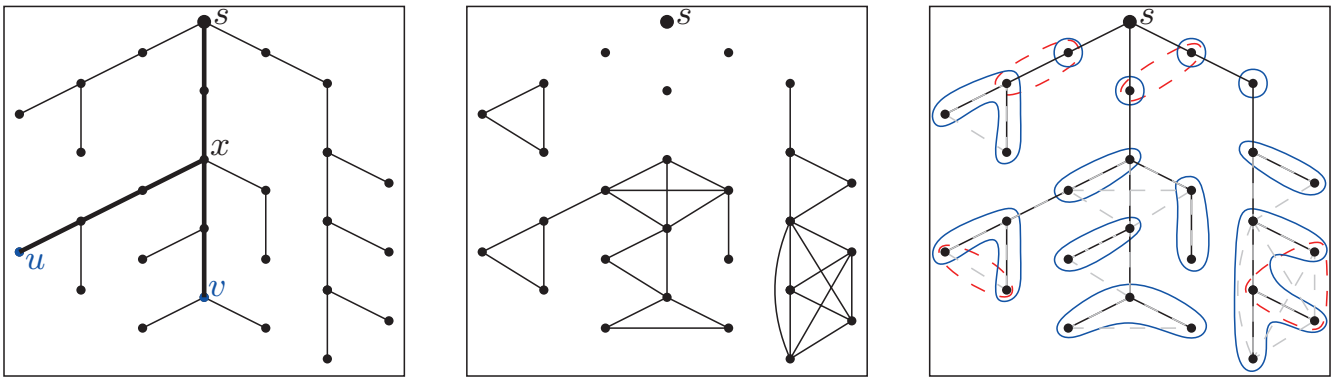
Throughout the paper, we let $G = (V,E)$ be an edge-weighted graph and $\mathcal{T} = (V,E_{\mathcal{T}})$ a spanning tree of G, rooted at a vertex $s \in V$. Given $\mathcal{T}$, we denote by P_{uv} the unique path in $\mathcal{T}$ that connects u and v; this is well-defined since $\mathcal{T}$ is a tree. Similarly, we write P_{uvw} for the concatenation of P_{uv} and P_{vw}; this notation extends to longer chains of paths. Given a path P, we denote the sum of weights of edges on the path by $w(P)$. Our results hold for arbitrary trees and arbitrary nonnegative edge weights, but in our application we specifically use shortest-path trees and Euclidean distances as weights.

Now we define a measure of similarity between two vertices by how much of their path to s they share: on the basis of this we will decide whether the vertices could reasonably be contracted in a nice visualization. Note that the following definition is not symmetric (see Figure 1 for illustrations of the following definitions).

Definition 1 (Directed Similarity) Let $u,v \in V$ be two vertices, $u \neq s$, and let x be their lowest common ancestor in $\mathcal{T}$. The directed similarity of u to v is defined as

$$\sigma(u,v) = w(P_{sx})/w(P_{su}). \tag{1}$$

The directed similarity is always between 0 and 1, inclusive, since paths have nonnegative weight and $w(P_{sx}) \leq w(P_{su})$ since x is on P_{su}.



(a) $\sigma(u,v) = \frac{2}{5}$ and $\sigma(v,u) = \frac{1}{2}$; hence u and v are α-compatible only for $\alpha \leq \frac{2}{5}$.

(b) Compatibility graph $G_\oplus$ for $\alpha = \frac{2}{3}$.

(c) Selection of allowed contractions (blue, computed with Algorithm 1) and counterexamples (red) due to a lack of connectivity in $\mathcal{T}$ or $G_\oplus$.

Figure 1: *Illustrations for Definitions 1 and 2, 3, and 4. All edges have weight* 1.

Definition 2 (α-Compatible, $\oplus$) Two vertices $u, v \in V$ are called α-compatible if and only if both

$$\sigma(u,v) \geq \alpha \quad \text{and} \quad \sigma(v,u) \geq \alpha. \tag{2}$$

When α is clear from context, we write $u \oplus v$ to assert that u and v are compatible.

Note that compatibility is reflexive and symmetric, but not transitive. For the rest of the paper we assume that a constant $\alpha \in [0,1]$ is used throughout.

Definition 3 (Compatibility graph) The *compatibility graph* $G_\oplus = (V, E_\oplus)$ has an edge between any two vertices $u, v \in V$ if and only if $u \oplus v$.

Definition 4 (Allowed contraction) Contracting a set of vertices $S \subseteq V$ is called *allowed* if and only if S is connected in $\mathcal{T}$ and all vertices in S are pairwise α-compatible. Note that the latter is equivalent to S being a clique in $G_\oplus$.

These definitions lead naturally to the following problem statement.

	TREESUMMARY
Instance:	A tree $\mathcal{T} = (V, E_\mathcal{T})$ rooted at $s \in V$.
	Weights on the edges $w : E_\mathcal{T} \to \mathbb{R}_{\geq 0}$.
	A compatibility threshold $\alpha \in [0,1]$.
	An integer k.
Question:	Does there exist a partition of V into at most k cells, such that each cell is an allowed contraction?

Since allowed contractions correspond to cliques, this problem asks for a minimum-cardinality clique cover of $G_\oplus$. (The proof of Theorem 1 will show that there exists an optimal clique cover where each cell is connected in $\mathcal{T}$.) The CLIQUECOVER problem on general graphs is one of Karp's original 21 $\mathcal{NP}$-complete problems [Kar72], but polynomial-time solvable for several restricted graph classes. (Sometimes the term CLIQUEPARTITION is used instead – interchangeably, it seems. In our situation we cover the vertices, not the edges.) In particular, we solve the optimization version of TREESUMMARY in linear time. First we give some structural lemmas. Consult Figure 2 for illustrations of the involved vertices and sets.

Lemma 1 (Compatible descendants) Let $u, v \in V$ be vertices and let x be their lowest common ancestor. Then u and v are compatible if and only if they are each compatible to x, that is, $u \oplus v \iff u \oplus x \wedge v \oplus x$.

Proof Implication both ways.

($\Longrightarrow$) Note that $\sigma(u,x) = \sigma(u,v)$ since x is the lowest common ancestor of u and v. By $u \oplus v$ we have $\sigma(u,v) \geq \alpha$ and therefore $\sigma(u,x) \geq \alpha$. The lowest common ancestor of u and x is in fact x, since u is a descendant of x: then $\sigma(x,u) = w(P_{sx})/w(P_{sx}) = 1 \geq \alpha$. This gives $u \oplus x$. The same argument for v gives $v \oplus x$.

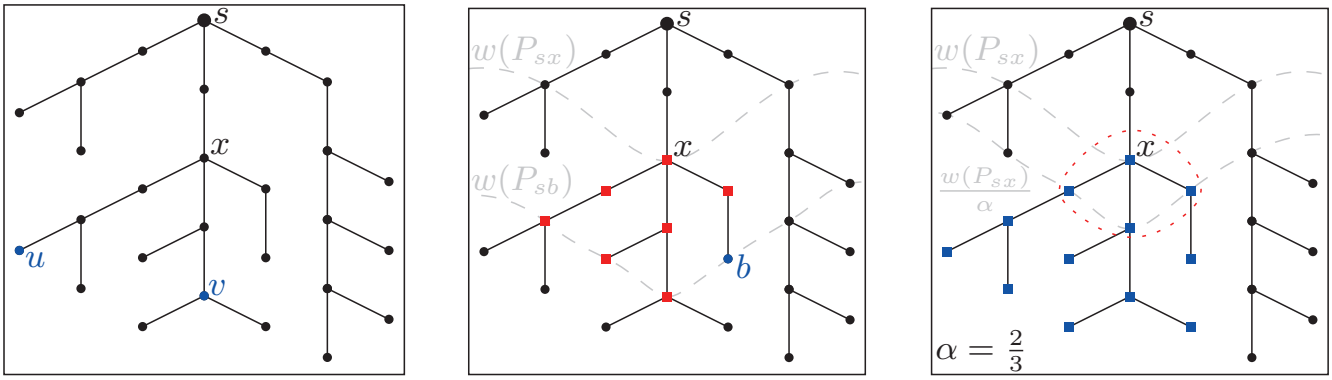
($\Longleftarrow$) Since $u \oplus x$, we have by definition that $\sigma(u,x) \geq \alpha$. Symmetrically $v \oplus x$ gives $\sigma(v,x) \geq \alpha$. Because x is the lowest common ancestor of u and v, this is precisely the definition of $u \oplus v$.

This concludes the proof. □

Lemma 2 (Compatible subtrees) Let $x \in V$ and let $a, b \in V$ be descendants of x. If $w(P_{sa}) \leq w(P_{sb})$ and $b \oplus x$, then $a \oplus x$. That is, if a and b are both descendants of x, and b is farther away from the root, then $b \oplus x \implies a \oplus x$.

Proof The lowest common ancestor of b and x is x itself, and we have $b \oplus x$, hence $w(P_{sx})/w(P_{sb}) \geq \alpha$. Since the lowest common ancestor of a and x is x as well, we have $\sigma(a,x) = w(P_{sx})/w(P_{sa}) \geq w(P_{sx})/w(P_{sb}) \geq \alpha$. With $\sigma(x,a) = 1 \geq \alpha$ we get $a \oplus x$. □

Lemma 3 (Equivalent descendants) Let $x \in V$ be a vertex and let $S \subseteq V$ be a set of vertices such that for any pair of vertices in S their lowest common ancestor is x. Then $(\oplus)$ is an equivalence relation on S. In particular, let $S_\oplus = \{v \in S : v \oplus x\}$. All pairs of vertices in $S_\oplus$ are compatible, and any vertex in $S \setminus S_\oplus$ is not compatible to any other vertex in S.



(a) The vertices u and v are compatible if and only if they are both compatible to x.

(b) If $b \oplus x$, then all square vertices (red) are compatible with x.

(c) In Lemma 3, S must be an appropriate subset of the square vertices; then the selected vertices encircled in red will form $S_\oplus$, all others are incompatible.

Figure 2: *Illustrations for Lemmas 1–3*

Proof Directly from Lemma 1. Let $u, v \in S$. If $u \oplus x$ and $v \oplus x$, then $u \oplus v$. If either $u \not\oplus x$ or $v \not\oplus x$, then $u \not\oplus v$. □

3. A linear-time algorithm for TREESUMMARY

We solve the optimization version TREESUMMARY: partitioning V into as few cells as possible, where every cell is an allowed contraction. The algorithm starts with every leaf vertex in a cell of its own and greedily merges cells in a post-order traversal (that is, bottom-up). During the algorithm, every cell C is an allowed contraction, that is, the following invariants hold:

(1) the vertices in C are connected in $\mathcal{T}$ and
(2) C is a clique in $G_\oplus$.

When the algorithm is done, the cells partition V and we will prove that the number of cells is minimized.

For a cell C, its vertex with the shortest path to s is called $Root(C)$. A vertex in C with the longest path to s (or tied for the longest) is called *deep*. Throughout the algorithm, we maintain the following references. Every vertex v, for as long as it is the root of a cell C, stores a reference $Cell(v) = C$. This is uniquely defined since cells are disjoint. Every cell C stores a reference $Deep(C)$ to any one of its deep vertices. These references can easily be kept up to date during the algorithm.

The algorithm traverses $\mathcal{T}$ in post-order. After processing a vertex, that vertex is the root of a cell. Therefore, when considering any vertex x, each child $c \in Children(x)$ is the root of a cell. We call a child c contractible if and only if $x \oplus Deep(Cell(c))$. We merge the cells of all contractible children – that is, we take their union – and add x to this cell: this results in a cell with x as its root. We do not do anything with incontractible children; their cells will not change anymore. See Algorithm 1.

This algorithm satisfies the two stated invariants. Upon initialization, both clearly hold. When processing a vertex, Lemma 3 holds for the deep vertices of its children's cells and then Lemma 2 holds for the entire cell. This shows that the (single) newly constructed cell is an allowed contraction.

Algorithm 1: ContractTree

Data: Rooted tree $\mathcal{T}$ with edge weights.
Result: Minimum-cardinality allowed contraction $\mathcal{C}$.

```
C ← the set with a singleton cell for each vertex of T;
forall the vertices x ∈ T, in post-order do
    S_⊕ ← {v ∈ Children(x) : x ⊕ Deep(Cell(v))};
    Merge Cell(x) and all Cell(v) for v ∈ S_⊕;
end
return C;
```

Theorem 1 Algorithm 1 computes an optimal solution and runs in $\mathcal{O}(n)$ time, where n is the number of vertices in $\mathcal{T}$.

Proof Let $\mathcal{C}$ be the set of cells determined by the algorithm. Note that $\mathcal{C}$ is a clique partition of $G_\oplus$ due to Lemma 3. Let $\mathcal{I} = \{Deep(C) : C \in \mathcal{C}\}$: we claim $\mathcal{I}$ is an independent set in $G_\oplus$. The tree $\mathcal{T}$ induces a tree structure on $\mathcal{C}$. Consider two sibling cells C_1 and C_2. If $Deep(C_1) \oplus Deep(C_2)$, then the algorithm would have merged these cells because of Lemma 1. Then, transitively, the fact that none of these cells were merged shows that all their deep vertices are mutually incompatible. Then $\mathcal{C}$ is an optimal clique partition of $G_\oplus$, since an independent set provides a lower bound for clique partitions and $|\mathcal{C}| = |\mathcal{I}|$.
During the tree traversal, each vertex considers all its children once. This involves testing $x \oplus Deep(Cell(v))$. If we precompute the weight $w(P_{sv})$ for each vertex v, this test runs in constant time. This bounds the runtime by $\mathcal{O}(n)$. □

Recall that a graph is called *chordal* if and only if all cycles of length at least 4 have a chord. Now we prove that the compatibility graph $G_\oplus$ is chordal. This immediately gives a polynomial-time algorithm for TREESUMMARY, since CLIQUECOVER can be solved in polynomial time on chordal graphs via its relation to coloring [Maf03]. We already have an easily-implementable linear-time algorithm based on the extra structure available to us (Algo-

rithm 1). Still, the chordality of $G_\oplus$ is interesting from a structural point of view.

Theorem 2 The compatibility graph $G_\oplus$ is chordal.

Proof Consider an arbitrary cycle in $G_\oplus$ with vertex set $V_C \subseteq V$ and $|V_C| \geq 4$. Let $u, v, w \in V_C$ be a sequence of vertices on this cycle where $\{u,v\}, \{v,w\} \in E_\oplus$ are edges of the cycle and v is the deepest vertex, that is, $w(P_{sv}) \geq w(P_{su})$ and $w(P_{sv}) \geq w(P_{sw})$. Furthermore let $x_u \in V$ be the lowest common ancestor of u and v (considering $\mathcal{T}$), and let $x_w \in V$ be the lowest common ancestor of w and v. Without loss of generality we assume $w(P_{sx_w}) \geq w(P_{sx_u})$.
In this situation, with both x_u and x_w being ancestors of v, either $x_u = x_w$ or x_w is a descendant of x_u. First assume $x_u = x_w$. With Lemma 1 we derive from $v \oplus u$ and $v \oplus w$ that both v and w are compatible with x_u. Then, another application of Lemma 1 tells us that $u \oplus w$ and thus $\{u,w\} \in E_\oplus$. Now assume that x_w is a descendant of x_u. Since $u \oplus v$ and Lemma 1 hold, u and v are compatible to their lowest common ancestor x_u. With w being a descendant of x_u and $w(P_{sv}) \geq w(P_{sw})$, the compatibility $v \oplus x_u$ results in $w \oplus x_u$ because of Lemma 2. Again, we have v and w being compatible with x_u. With Lemma 1 this results in $u \oplus w$ and thus $\{u,w\} \in E_\oplus$. We see in either case that V_C has a chord. Consequently, $G_\oplus$ is chordal. □

4. Map generalization

Now we return to map generalization and our example problem of finding a way to Groningen. Here, the TREESUMMARY problem does not directly apply: the road network we are interested in is unlikely to be a tree. Instead, we have a geometric graph $G = (V,E)$ with the user's position $s \in V$ and a nonnegative weight function $w\colon E \to \mathbb{R}_{\geq 0}$. (In order to generate destination maps [KAB*10] instead, we can pick s to be the destination.) In this section, we will first consider a tree in G, so that we may apply our tree contraction algorithm. Afterward we reintroduce the connectivity information from G in order to generate the generalized map.

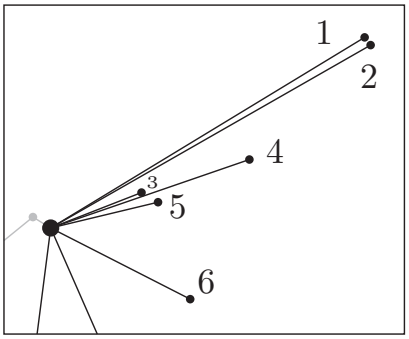
Since the generalized map is intended for planning routes, a shortest-path tree of G (rooted at s) seems appropriate. However, if we simply calculate a normal shortest-path tree, we lose a lot of information. Consider an edge $e = \{u,v\} \in E$, where neither the shortest path to u passes through v nor the other way around. This edge would not be included in a normal shortest-path tree. This is correct if we only consider the vertices of the graph. However, if the graph represents a continuous road network, then there is in fact a point on the interior of the edge at which s is equidistant through u and through v; anywhere else on the edge, one path or the other is shorter.

In order to incorporate this connectivity information in our visualization of the road network, we start by calculating a modified shortest-path tree $\mathcal{T}' = (V', E')$ where $V \subseteq V'$. This tree $\mathcal{T}'$ contains all edges from the normal shortest-path tree, but as noted, there are edges in E that do not show up. For every such "missing" edge $e = \{u,v\}$ we do the following: we find the position on e where the paths to s via u and via v have the same length, where we assume the weight of e is distributed uniformly along its length. (This holds, of course, if the weights are Euclidean distances. If, for another example, the weights represent travel time, the assumption is still reasonable.) At this position on e, we introduce two *virtual vertices* p_1 and p_2 to V' and add the edges $\{u,p_1\}$ and $\{p_2,v\}$ to E', assigning weights appropriately. (To contrast these virtual vertices, the original vertices of V are called *real*.) By annotating p_1 and p_2 with references to the other, the resulting tree $\mathcal{T}'$ represents all edges and connectivity of G.

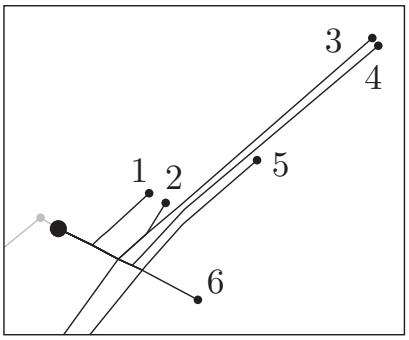
After constructing $\mathcal{T}'$, we apply Algorithm 1 to solve TREESUMMARY on it. The result is a partition of V' into contractible cells. These cells are highly detailed close to s and less detailed the farther away they are. The question remains how to draw a generalized map based on these cells.

We represent each cell by its root vertex, which we draw in its original position given by G. Note that the cells themselves are also related in a tree structure induced by $\mathcal{T}'$: call a cell C the child of another cell P if and only if $Root(P)$ is the first cell root encountered on the path from $Root(C)$ up to the tree root s. We propose the following three visualizations of $\mathcal{T}'$.

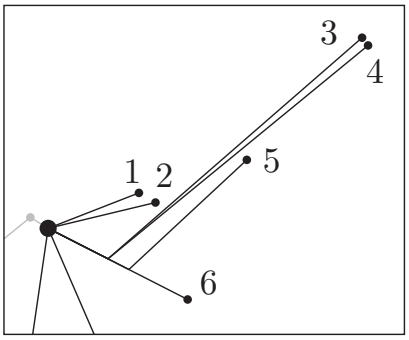
- For every cell C and its parent cell P, let $c = Root(C)$, $p = Root(P)$ and draw a straight line segment between c and p. We call this the *direct* drawing. The advantage of this drawing is that it is simple, highly generalized, and has a one-to-one correspondence between cells and output vertices. We do note that in general these line segments may intersect and that, even if they do not, the embedding of the output map may be inconsistent with the input. (See Figures 3 and 4(b) for an illustration.) This poses problems for a road map: intersecting line segment may visually imply connectivity, and even if the appropriate details will show up once the user comes closer, an incorrect embedding may negatively impact the user's mental model.
- For every cell C and its parent cell P, let $c = Root(C)$, $p = Root(P)$ and draw P_{cp}. That is, draw the path in $\mathcal{T}'$ that connects c and p. We call this the *detailed* drawing. Note that P_{cp} is an actual shortest path in G. This is a positive aspect of the detailed drawing: it is in fact an edge selection of G and it contains those parts of the shortest path tree that connect the cell roots. This does mean that, even though a selection has been made, the selected polylines still have the same level of detail as the input, which may or may not be desirable.
  It should also be noted that this drawing can have vertices of degree larger than two that are not at a cell root: we call these *internal branches*. (See Figures 3 and 4 for an illustration.) The existence of such vertices can be considered somewhat unfortunate after we have carefully optimized the selection of cell roots, but this drawing is true to G and it does not necessarily look displeasing.
- Construct the detailed drawing and apply a topologically-safe simplification algorithm. We call this the *simplified* drawing. We use the algorithm of Dyken et al. [DDS09], which greedily deletes vertices but never moves them, and never deletes cell roots. We run the algorithm without stopping criterion, that is, as far as it will go. In order to (hopefully) get rid of any internal branches that the detailed drawing might have, we slightly offset each path so that they do not actually touch except in cell roots and therefore do not actually form internal branches. (Topologically, this can always be done and it can be implemented simply in practice if one accepts some inefficiencies. It has not been our



(a) Direct.



(b) Detailed.

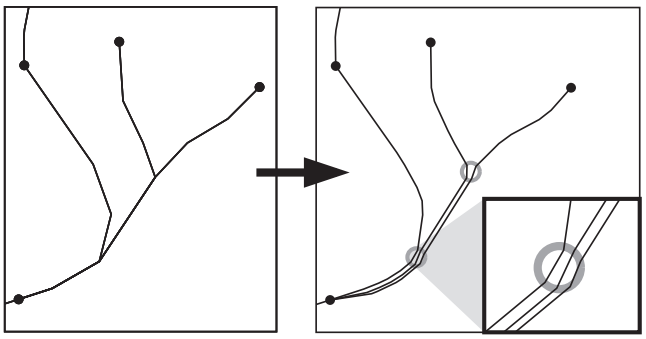


(c) Simplified.

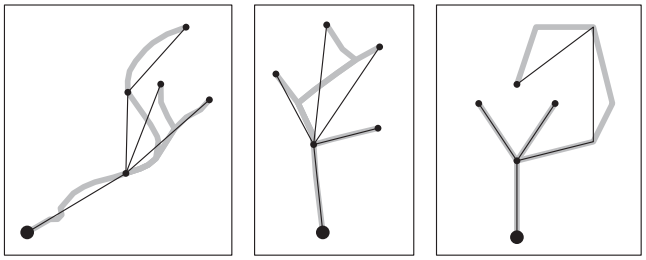
Figure 3: *Using direct line segments between cell roots is not topologically safe. Even if there are no intersections in the output, it may still be the case that the embedding changes as illustrated in this example: the children of the fat vertex are numbered in clockwise order.*

goal to optimize this step.) We call this the peeled tree: see Figure 4(a). After simplification we merge any remaining vertices that logically belong to the same vertex in V'. This may introduce internal branches, but ensures that the final drawing does not contain infinitesimally-shifted paths

Figure 5 shows a map of the German city of Würzburg, and *detailed* and *simplified* drawings for the same root s. It can be seen that the detailed drawing already provides a focus-and-context effect, giving more detail near the bottom of the map, slightly left of the middle, which is where s is located. When compared to the



(a) Peeling a tree.



(b) Some simplifications. Fat gray lines indicate the input; black lines indicate the output.

Figure 4: *Simplifying a tree as a set of polylines. Dots indicate cell roots.*

simplified drawing, it can be seen that internal branches are present. Simplifying those gives a much cleaner output graph, but naturally this comes at some cost to the recognizability of the road network. In this particular example, only two internal branches remain in the simplified drawing.

When drawing maps, restricting ourselves to outputting trees limits the information we can present. Indeed, as we argued at the start of the section, we should care about the cross links between the cells: between cells that share a virtual vertex, to be precise, since these are cells that touch in the network G. This occurs on the interior of edges in E (or in vertices of V) and it is actually possible to navigate from a cell to a touching cell. We now draw these adjacencies, either *directly*, with a straight-line segment, or with a *detailed* path: from one cell root, to the virtual vertex, and then to the other cell root. If a pair of cells touch in more than one virtual vertex, we draw this connection only for a single virtual vertex that has the shortest distance to s.

Figure 6 shows the resulting drawing of Würzburg, for varying values of α. More so than the tree drawing of Figure 5, this resembles a traditionally useful map and it also exhibits a focus-and-context effect, the strength of which depends on α.

Figure 7(a) shows a map of a much larger road network (Dallas, Texas). Figure 7(b) shows a *detailed* drawing, with cross connections and $\alpha = 0.5$. This already gives a focus-and-context effect by having full detail in an area of interest and a gradually more generalized road selection as the (network) distance increases. Figure 7(c) shows an example of how this effect can be enhanced and

(a) Input. (b) Detailed. (c) Simplified.

Figure 5: *Würzburg with $\alpha = 0.8$, with the vertex s indicated in figure (a). The simplified drawing contains only two internal branches: Figure 3 is a crop of this map showing the extra vertices (located near the bottom right of this figure).*

makes the map more useful by computing a variable-scale map transformation in the style of Haunert [HS11, vDvGH*13].

We finish with an observation about the abovementioned cross connections. In order to provide meaningful information to the user, we want to give an estimation of how long a detour might get if the user chooses to use the shortest path to a neighboring cell and such a cross link instead of following the route on $\mathcal{T}'$.

Theorem 3 A detour through a neighboring cell is at most $\left(\frac{2}{\alpha} - 1\right)$ times as long as the shortest path.

Proof Consider an arbitrary vertex $u \in V'$, either real or virtual. Let $v \in V' \setminus V$ be any virtual vertex in the same cell as u (possibly $u = v$) and let $r \in V$ be the root of the cell containing u. Then the detour of going through v is a factor $\delta := w(P_{uvs})/w(P_{us})$. We wish to bound δ.
We first observe by the triangle inequality for shortest paths that the distance from u to v is at most the distance from u to v via r. This detour is possible since u and v are both in the cell with root r. This gives

$$1 \leq \delta \leq \frac{w(P_{urvs})}{w(P_{us})} = \frac{w(P_{ur}) + w(P_{rv}) + w(P_{vs})}{w(P_{ur}) + w(P_{rs})}.$$

Since all the weights are nonnegative and u influences the right-hand side only through $w(P_{ur})$, we can maximize the given ratio by setting $w(P_{ur}) = 0$, since $w(P_{ur})$ appears in both the numerator and the denominator as a summand. (That is, the detour is worst when u is a cell root, which makes sense.)
We observe that $w(P_{rv}) = w(P_{vs}) - w(P_{rs})$ because r lies on P_{vs}. Since v and its ancestor r lie in the same cell, they are compatible and we have $w(P_{vs}) \leq \frac{1}{\alpha} w(P_{rs})$. Putting these things together,

$$\begin{aligned}\delta &\leq \frac{w(P_{rv}) + w(P_{vs})}{w(P_{rs})} = \frac{\big(w(P_{vs}) - w(P_{rs})\big) + w(P_{vs})}{w(P_{rs})} \\ &\leq \frac{\left(\left(\frac{1}{\alpha} - 1\right) + \frac{1}{\alpha}\right) \cdot w(P_{rs})}{w(P_{rs})} = \frac{2}{\alpha} - 1.\end{aligned}$$

This is the bound in the theorem. □

For $\alpha = 0.95$, as proposed in the introduction, the above theorem gives an upperbound of 11% on any detour, which seems very reasonable. The value $\alpha = 0.9$ results in 22% and even for $\alpha = 0.5$ we still get a factor 3.

Experimental setup

We have implemented Algorithm 1 in *C++*. To generate the simplified drawings, we have used CGAL's implementation [Fab15] of a topologically-safe simplification algorithm by Dyken et al. [DDS09]. The computations for this paper were run on a desktop PC with an Intel® Core™ i5-2400 CPU at 3.10 GHz; memory usage was not an issue. The map of Würzburg is a crop of the *OpenStreetMap* road network of Würzburg, Germany (`http://download.geofabrik.de`). It uses Euclidean weights and contains approximately 2500 vertices. The runtime of both Algorithm 1 and the simplification is instant. The map of Dallas is the largest connected component in the *City of Dallas GIS Services*' road network of Dallas, Texas (`http://gis.dallascityhall.com/EnterpriseGIS`). It has approximately 3×10^5 vertices and the weights are a travel-time estimate based on road class. Algorithm 1 still runs almost instantly, given $\mathcal{T}'$. The entire computation, which includes building $\mathcal{T}'$ using an unoptimized implementation and the simplification, has a runtime of about 2 seconds. We conclude that this approach is suitable for interactive applications on realistically-sized maps.

(a) Detailed, $\alpha = 0.3$. (b) Detailed, $\alpha = 0.5$. (c) Detailed, $\alpha = 0.7$.

(d) Detailed, $\alpha = 0.9$. (e) Direct, $\alpha = 0.5$. (f) Direct, $\alpha = 0.7$.

Figure 6: *Würzburg with cross connections, for various values of α, and with either detailed of direct drawing style. All drawings use the same vertex s, which is indicated in figure (a). Note that figures (b) and (e) are the detailed and the direct drawing of the same underlying partition; this also holds for (c) and (f).*

5. Conclusion and Outlook

We have proposed partitioning the vertices of a tree based on whether or not they share a significant part of their path to the root. We gave a linear-time algorithm for minimizing the number of cells in this partition and proved several properties of such clusterings. This approach could be used to model various clustering problems on trees, such as summarizing large hierarchies (organizational charts, file systems, et cetera). We looked specifically at a map generalization problem where we summarize a shortest-path tree and were able to derive a constant-factor bound on the "generalization error" as measured by the detours resulting from using the output map.

We would argue that our visualization of the computed clustering looks promising, but is certainly open for improvement. A more elegant solution to the topological issue of internal branches would be welcome. After all, we have taken care to compute an optimal solution to the TREESUMMARY problem: we have already decided what we *want* to draw. Given a desire for topological safety and without moving any vertices, we may be forced to include additional detail. Letting go of exact geographic positions may lead to better results by allowing map deformations. What deformations are appropriate, and how to compute them, would clearly depend on the application. Another direction would be a visualization where the clusters are represented by areal features.

Other map generalization scenarios present themselves depending on which information is available to us. Say we are not only given the weighted graph $G = (V, E)$, but additionally get a partition $\mathcal{P} = \{V_1, \ldots, V_k\}$ of the vertex set, where each cell of $\mathcal{P}$ reflects a geographic region (such as "Groningen"). The task is then to decide for each cell $V_i \in \mathcal{P}$ whether or not the vertices in V_i

(a) Input network. (b) Detailed drawing, $\alpha = 0.5$. (c) Focus map.

Figure 7: *The city of Dallas, Texas. In Figure (b), the black lines indicate a detailed drawing of the tree summary and the gray lines indicate cross connections. The focus map (c) was computed with the method of Van Dijk et al. [vDvGH*13], using a scale factor of 3 for a region around the selected center.*

are equivalent, for which we might use our contractibility condition. Depending on the result, V_i will be collapsed and displayed as a single vertex or each vertex in V_i will be displayed individually. One can also consider the case where a hierarchical partition is given (neighborhoods, cities, states, federations). An advantage of this setting is that the cells are likely to have meaningful names, which will help when labeling the map. This raises the open question of how labels for geographic regions can be carried over to a clustering computed using our algorithm. As a disadvantage of the pre-clustered approach, it is more constrained in its options than our approach, so it will use at least as many clusters and probably more.

Another direction for future work is to consider a dynamic version of the problem, where the vertex s changes. This is relevant, for example, when using this visualization in an in-car navigation system. Our algorithms are fast enough to be run in realtime on realistic maps, but this does not close the case: for interactive or animated visualizations, it is important to consider the stability of the computed solutions. This should be considered as an optimization problem, but is currently open.

References

[AS01] AGRAWALA M., STOLTE C.: Rendering effective route maps: Improving usability through generalization. In *Proc. 28th Conference on Compututer Graphics and interactive techniques* (2001), pp. 241–249. doi:10.1145/383259.383286. 2

[BGK*14] BRUNEL E., GEMSA A., KRUG M., RUTTER I., WAGNER D.: Generalizing geometric graphs. *Journal of Graph Algorithms and Applications 18*, 1 (2014), 35–76. doi:10.7155/jgaa.00314. 2

[BL07] BLANCH R., LECOLINET E.: Browsing zoomable treemaps: Structure-aware multi-scale navigation techniques. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1248–1253. 2

[BYB*13] BORKIN M. A., YEH C. S., BOYD M., MACKO P., GAJOS K. Z., SELTZER M., PFISTER H.: Evaluation of filesystem provenance visualization tools. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2476–2485. 2

[CKB09] COCKBURN A., KARLSON A., BEDERSON B. B.: A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys 41*, 1 (2009), 2:1–2:31. doi:10.1145/1456650.1456652. 2

[CvDH14] CHIMANI M., VAN DIJK T. C., HAUNERT J.-H.: How to eat a graph: Computing selection sequences for the continuous generalization of road networks. In *Proc. 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM-GIS'14)* (2014), pp. 243–252. doi:10.1145/2666310.2666414. 2

[DDS09] DYKEN C., DÆHLEN M., SEVALDRUD T.: Simultaneous curve simplification. *Journal of Geographical Systems 11*, 3 (2009), 273–289. doi:10.1007/s10109-009-0078-8. 1, 6, 7

[Fab15] FABRI A.: 2D polyline simplification. In *CGAL User and Reference Manual*, 4.6 ed. CGAL Editorial Board, 2015. URL: http://doc.cgal.org/4.6/Manual/packages.html#PkgPolylineSimplification2Summary. 7

[Fur86] FURNAS G. W.: Generalized fisheye views. In *Proc. ACM Conference on Human Factors in Computing Systems (SIGCHI'86)* (1986), pp. 16–32. 2

[GSSV12] GEISBERGER R., SANDERS P., SCHULTES D., VETTER C.: Exact routing in large road networks using contraction hierarchies. *Transportation Science 46*, 3 (2012), 388–404. doi:10.1287/trsc.1110.0401. 2

[HRR*07] HUSON D. H., RICHTER D. C., RAUSCH C., DEZULIAN T., FRANZ M., RUPP R.: Dendroscope: An interactive viewer for large phylogenetic trees. *BMC Bioinformatics 8*, 1 (2007), 1–6. URL: http://dx.doi.org/10.1186/1471-2105-8-460, doi:10.1186/1471-2105-8-460. 2

[HS11] HAUNERT J.-H., SERING L.: Drawing road networks with focus regions. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2555–2562. 6

[HW10] HAUNERT J.-H., WOLFF A.: Area aggregation in map generalisation by mixed-integer programming. *International Journal of*

Geographical Information Science 24, 12 (2010), 1871–1897. doi:10.1080/13658810903401008. 1

[JC04] JIANG B., CLARAMUNT C.: A structural approach to the model generalization of an urban street network. *Geoinformatica 8*, 2 (2004), 157–171. doi:10.1023/B:GEIN.0000017746.44824.70. 2

[KAB*10] KOPF J., AGRAWALA M., BARGERON D., SALESIN D., COHEN M.: Automatic generation of destination maps. *ACM Transactions on Graphics 29*, 6 (dec 2010), 158:1–158:12. doi:10.1145/1882261.1866184. 2, 5

[Kar72] KARP R.: Reducibility among combinatorial problems. In *Complexity of Computer Computations*, Miller R. E., Thatcher J. W., Bohlinger J. D., (Eds.), The IBM Research Symposia Series. Springer US, 1972, pp. 85–103. doi:10.1007/978-1-4684-2001-2_9. 3

[LRP95] LAMPING J., RAO R., PIROLLI P.: A focus + context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proc. ACM Conference on Human Factors in Computing Systems (SIGCHI'95)* (1995), ACM Press/Addison-Wesley Publishing Co., pp. 401–408. 2

[Maf03] MAFFRAY F.: On the coloration of perfect graphs. In *Recent Advances in Algorithms and Combinatorics*, Reed B. A., Linhares-Sales C. L., (Eds.), CMS Books in Mathematics / Ouvrages de mathématiques de la SMC. Springer New York, 2003, pp. 65–84. doi:10.1007/0-387-22444-0_3. 4

[MB93] MACKANESS W. A., BEARD M. K.: Use of graph theory to support map generalization. *Cartography and Geographic Information Systems 20* (1993), 210–221. URL: http://www.geos.ed.ac.uk/homes/wam/MackBeard1993.pdf. 1

[Ses05] SESTER M.: Optimization approaches for generalization and data abstraction. *International Journal of Geographical Information Science 19*, 8–9 (2005), 871–897. doi:10.1080/13658810500161179. 1

[SS12] SANDERS P., SCHULTES D.: Engineering highway hierarchies. *ACM Journal of Experimental Algorithmics 17*, 1 (2012). doi:10.1145/2133803.2330080. 2

[TB02] THOMSON R. C., BROOKS R.: Exploiting perceptual grouping for map analysis, understanding and generalization: The case of road and river networks. In *Selected Papers from the Fourth Internat. Workshop Graphics Recognition Algorithms and Applications (GREC '01)* (2002), vol. 2390 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, Germany, pp. 148–157. doi:10.1007/3-540-45868-9_12. 2

[TP66] TÖPFER F., PILLEWIZER W.: The principles of selection. *The Cartographic Journal 3*, 1 (1966), 10–16. 1

[vDH14] VAN DIJK T. C., HAUNERT J.-H.: Interactive focus maps using least-squares optimization. *International Journal of Geographical Information Science 28*, 10 (2014), 2052–2075. doi:10.1080/13658816.2014.887718. 2

[vDvGH*13] VAN DIJK T. C., VAN GOETHEM A., HAUNERT J.-H., MEULEMANS W., SPECKMANN B.: Accentuating focus maps via partial schematization. In *Proc. 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2013), ACM, pp. 428–431. 6, 9

[VvWvdL06] VLIEGEN R., VAN WIJK J. J., VAN DER LINDEN E.-J.: Visualizing business data with generalized treemaps. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sept 2006), 789–796. doi:10.1109/TVCG.2006.200. 2

[WJT03] WARE J. M., JONES C. B., THOMAS N.: Automated map generalization with multiple operators: a simulated annealing approach. *International Journal of Geographical Information Science 17*, 8 (2003), 743–769. doi:10.1080/1365881031000159608 5. 1

[YOT09] YAMAMOTO D., OZEKI S., TAKAHASHI N.: Focus+glue+context: An improved fisheye approach for web map services. In *Proc. 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM-GIS'09)* (2009), pp. 101–110. doi:10.1145/1653771.1653788. 2

[Zha05] ZHANG Q.: Road network generalization based on connection analysis. In *Developments in Spatial Data Handling – Proc. 11th Internat. Sympos. Spatial Data Handling (SDH '04)* (2005), Springer-Verlag, Berlin, Germany, pp. 343–353. doi:10.1007/3-540-26772-7_26. 1

[ZR02] ZIPF A., RICHTER K.-F.: Using focus maps to ease map reading: Developing smart applications for mobile devices. *Künstliche Intelligenz 02*, 4 (2002), 35–37. 2