Using Graph Layout to Visualize Train Interconnection Data

Ulrik Brandes and Dorothea Wagner

University of Konstanz
Faculty of Mathematics and Computer Science
Fach D 188, D–78457 Konstanz, Germany
{Ulrik.Brandes, Dorothea.Wagner}@uni-konstanz.de

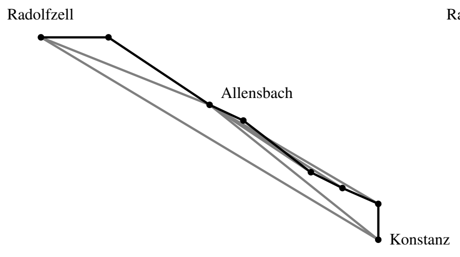
Abstract. We are concerned with the problem of visualizing interconnections in railroad systems. The real-world systems we have to deal with contain connections of thousands of trains. To visualize such a system from a given set of time tables a so-called train graph is used. It contains a vertex for each station met by any train, and one edge between every pair of vertices connected by some train running from one station to the other without halting in between.
In visualizations of train graphs, positions of vertices are predetermined, since each station has a given geographical location. If all edges are represented by straight-lines, the result is visual clutter with many overlaps and small angles between pairs of lines. We here present a non-uniform approach using different representations for edges of distinct meaning in the exploration of the data. Only edges of certain type are represented by straight-lines, whereas so-called transitive edges are rendered using Bézier curves. The layout problem then consists of placing control points for these curves. We transform it into a graph layout problem and exploit the generality of random field layout models for its solution.

1 Introduction

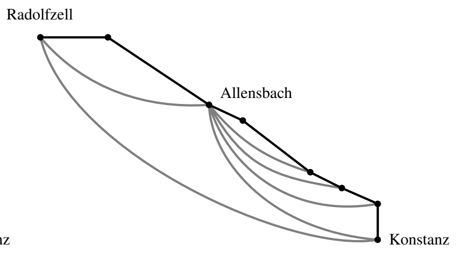
The layout problem we are concerned with arises from a cooperation with a subsidiary of the *Deutsche Bahn AG* (the central German train and railroad company), *TLC/EVA*. The aim of this cooperation is to develop data reduction and visualization techniques for the explorative analysis of large amounts of time table data from European public transport systems. For the most part, these are comprised of train schedules. However, the data may also contain bus, ferry and footwalk connections. The analysis of the data with respect to completeness, consistency, changes between consecutive periods of schedule validity, and so on is relevant, e.g., for quality control, (international) coordination, and pricing.

To condense the data, a *train graph* is built in the following way: For each regular stop of any train, a vertex is added to the network. One arc is added, if there is service from one station to another without intermediate stops. For convenience, we assume that for each train operating between two stations, there is a corresponding train serving the opposite direction. Hence, the train graphs considered here are simple and undirected.

An important aspect is the classification of edges in two categories: *minimal* edges and *transitive* edges. Minimal edges are those corresponding to a set of continuous connections between two stations not passing through a third one. Typically, these are induced by regional trains stopping at every station. On the other hand, transitive edges correspond to connections passing through other stations without halting. These are induced by through-trains. Figure 1(a) shows part of a train graph with edges colored according to this classification. Stations are positioned according to their geographical location, and all edges are represented straight-line. An obvious problem are edge overlaps and small angles between edges. In order to maintain geographic familiarity, we are not allowed to move vertices. Since minimal edges usually represent actual railways, they also remain the same, but we refrain from drawing all transitive edges straight-line. Instead, we use Bézier curves as shown in Fig. 1(b).



(a) straight lines



(b) Bézier curves

Fig. 1. Two different representations of transitive edges in a small train graph

To render Bézier curves, control points need to be positioned. Using the framework of random field layout models introduced in [5], the problem is cast into a graph layout problem. More precisely, we consider control points to be vertices of a graph, and rules for appropriate positioning are modeled by defining edges accordingly. This way, common algorithmic approaches can be employed. Practical applicability of our approach is gained from experimental validation. In a completely different field of application, the same strategy is used to identify suitable layout models for social and policy networks [3]. These real-world applications are good examples of how the uniform approach of random field layout models may be used to obtain initial models for visualization problems which are not clearly defined beforehand.

The paper is organized as follows. In Sect. 2, we review briefly the concept of random field layout models. A specific random field model for train graph layout is defined in Sect. 3. Section 4 contains experiments with real-world examples and a short discussion on aspects of parametrization.

2 Random Field Models

In this section we review briefly the uniform graph layout formalism introduced in [5]. As can be seen from Section 3, model definition within this framework is straightforward.

Virtually every graph layout problem can be viewed as a constrained optimization problem. A layout of a graph $G = (V, E)$ is computed by assigning values to certain layout variables, subject to constraints and an objective function. Straight-line embeddings, for example, are completely determined by an assignment of coordinates to each vertex. But straight-line representations are only a very special case of a layout problem. In its most general form, each element of a set $L = \{l_1, \ldots, l_k\}$ of arbitrary *layout elements* is assigned a value from a set of allowable values $\mathcal{X}_l$, $l \in L$. Layout elements may represent positional variables for vertices, edges, labels, and any other kind of graphical object. Therefore, L and $\mathcal{X} = \mathcal{X}^L = \mathcal{X}_{l_1} \times \cdots \times \mathcal{X}_{l_k}$ are clearly dependent on the chosen type of graphical representation. In this application, we do not constrain configurations of layout elements. Hence, all vectors $x \in \mathcal{X}$ are considered feasible *layouts*.

Objective Function. In order to measure the quality of a layout, an objective function $U : \mathcal{X} \to \mathrm{I\!R}$ is defined. It is based on configurations of subsets of layout elements which mutually influence their positioning. This interaction of layout elements is modeled by an *interaction graph* $G^\eta = (L, E^\eta)$ that is obtained from a *neighborhood system* $\eta = \bigcup_{l \in L} \eta_l$, where $\eta_l \subseteq L \setminus \{l\}$ is the set of layout elements for which the position assigned to l is relevant in terms of layout quality. These interactions are symmetric, i.e. $l_2 \in \eta_{l_1} \Leftrightarrow l_1 \in \eta_{l_2}$ for all $l_1, l_2 \in L$, so G^η is undirected. The set of cliques in G^η is denoted by $\mathcal{C} = \mathcal{C}(\eta)$. We define the *interaction potential* of a clique $C \in \mathcal{C}$ to be any function $U_C : \mathcal{X} \to \mathrm{I\!R}$ for which

$$x_C = y_C \quad \Rightarrow \quad U_C(x) = U_C(y)$$

holds for all $x, y \in \mathcal{X}$, where $x_C = (x_l)_{l \in C}$. A graph layout objective function $U : \mathcal{X} \to \mathrm{I\!R}$ is the sum of all interaction potentials, i.e. $U(x) = \sum_{C \in \mathcal{C}} U_C(x)$. By convention, the objective function is to be minimized. $U(x)$ is often called the *energy* of x.

Fundamental Potentials. One advantage of separating the energy function into clique potentials is that recurrent design principles can be isolated to form a toolbox of fundamental potentials. Not surprisingly, the two most basic potentials are those corresponding to the forces used in the spring embedder [7]:[1]

– *Repelling Potential:* The rule that two layout elements k and l should not lie close to each other can be expressed by a potential

$$U^{(rep)}_{\{k,l\}}(x) = Rep(x_k, x_l) = \frac{\varrho}{d(x_k, x_l)^2}$$

[1] The original spring embedder does not specify an objective function, but its gradients. The above potentials appear in [6].

where ϱ is a fixed constant and $d(x_k, x_l)$ is the Euclidean distance between the positions of k and l. $Rep(x_k, x_l \mid \varrho)$ is used to indicate a specific choice of ϱ.

– *Attracting Potential:* If, in contrast, k and l should lie close to each other, a potential

$$U^{(attr)}_{\{k,l\}}(x) = Attr(x_k, x_l) = \alpha \cdot d(x_k, x_l)^2,$$

with α a fixed constant, is appropriate. Like above we use $Attr(x_k, x_l \mid \alpha)$ to denote a specific choice of α.

Since $Rep(x_k, x_l \mid \lambda^4) + Attr(x_k, x_l \mid 1)$ is minimized when $d(x_k, x_l) = \lambda$, it is easy to specify a desired distance between two layout elements (e.g. edge length). Note that many other design rules (sufficiently large angles, vertex-edge distance, edge crossings, etc.) are easily formulated in terms of clique potentials [5].

If layouts $x \in \mathcal{X}$ are assigned probabilities

$$P(X = x) = \frac{1}{Z} e^{-U(x)},$$

where $Z = \sum_{y \in \mathcal{X}} e^{-U(y)}$ is a normalizing constant, random variable X is a (Gibbs) random field. Both X and its distribution are called a (random field) *layout model* for G. Clearly, the above probabilities depend on the energy only, with a layout of low energy being more likely than a layout of high energy. By using a random variable, the entire layout model is described in a single object. Due to the familiar form of its distribution, a wealth of theory becomes applicable (a primer in the context of dynamic graph layout is [4]). See [11] for an overview on the theory of random fields, and some of its applications in image processing. Since random fields are used so widely, there also is a great deal of literature on algorithms for energy minimization (see e.g. [10]).

3 Layout Model

We now define a random field model for the layout of a train graph $G = (V, E)$. Vertex positions are given by geographical locations of corresponding stations, and minimal edges as well as very long transitive edges are represented straight-line. For the other edges we use Bézier cubic curves (cf. Fig. 2).[2] Let $\hat{E} \subseteq E$ be the set of transitive edges of length less than a threshold parameter τ_1, such that the set of layout elements consists of two control points for each edge in $\hat{E}$, $L = \{b_u(e), b_v(e) \mid e = \{u, v\} \in \hat{E}\}$. If two Bézier points belong to the same edge, they are called *partners*. The *anchor*, $a_{b_u(e)}$, of $b_u(e)$ is u, while the anchor of $b_v(e)$ is v. The *default position* of all Bézier points is on the straight line through the endpoints of their edges at equal distance from their anchor and from their partner (and hence uniquely defined).

[2] It will be obvious from the examples presented in Section 4 why it is not useful to represent all transitive edges by Bézier curves.

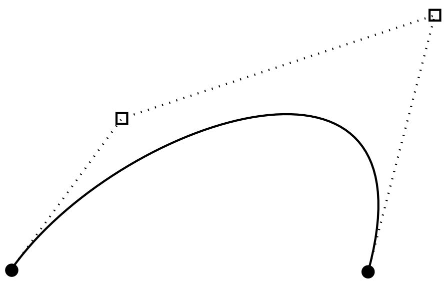


Fig. 2. Bézier cubic curve [2]. Two endpoints and two control points define a smooth curve that is entirely enclosed by the convex hull of these four points

The position assigned to a Bézier point is influenced by its partner, its anchor, all Bézier points with the same anchor, and a number of close stations and Bézier points. Let $\{u, v\} \in \hat{E}$ be a transitive edge, and let $b \in L$ be a Bézier point of $\{u, v\}$. Given two parameters ϵ_1 and ϵ_2, consider an ellipse with major axis going through u and v. Let its radii be $\epsilon_1 \cdot \frac{d(u,v)}{2}$ and $\epsilon_2 \cdot \frac{d(u,v)}{2}$, respectively. We denote the set of all stations and Bézier points (at their default position) within this ellipse, except for b itself, by $\mathcal{E}_b$. Recall that the neighborhood of some layout element consists of all those layout elements that have an influence on its positioning. Therefore, η_b equals the union of $\mathcal{E}_b \cap L$, the set of Bézier points with the same anchor as b, and (since interactions need to be symmetric) the set of Bézier points b' for which $b \in \mathcal{E}_{b'}$. For the examples presented in Section 4 we used $\epsilon_1 = 1.1$ and $\epsilon_2 = 0.5$.

An interaction potential is defined for each design goal that a good layout of Bézier points should achieve:

– *Distance to stations.* For each Bézier point $b \in L$ of some edge $\{u, v\} \in \hat{E}$, there are repelling potentials

$$\sum_{s \in \mathcal{E}_b \cap V} \mathit{Rep}\left(x_b, s \mid (\varrho_1 \cdot \lambda_b)^4\right),$$

with $\lambda_b = \frac{d(u,v)}{3}$ and ϱ_1 a constant. These ensure reasonable distance from stations in the vicinity of b and can be controlled via ϱ_1. A combined repelling and attracting potential

$$\mathit{Rep}\left(x_b, a_b \mid (\lambda_1 \cdot \lambda_b)^4\right) + \mathit{Attr}(x_b, a_b),$$

where λ is another constant, keeps b sufficiently close to its anchor a_b.

– *Distance to near Bézier points.* As is the case with near stations, a Bézier point $b_1 \in L$ should not lie too close to another Bézier point $b_2 \in \eta_{b_1}$. If b_1 is neither the partner of nor bound to b_2 (binding is defined below), we add

$$\mathit{Rep}\left(x_{b_1}, x_{b_2} \mid \varrho_2^4 \cdot \min\{\lambda_{b_1}^4, \lambda_{b_2}^4\}\right).$$

The desired distance between partners b_1 and b_2 is equal to the desired distance from their respective anchors,

$$Rep\left(x_{b_1}, x_{b_2} \mid (\lambda_1 \cdot \lambda_{b_1})^4\right) + Attr(x_{b_1}, x_{b_2}).$$

– *Binding.* In general, it is not desirable to have Bézier points $b_1, b_2 \in L$ with a common anchor lie on different sides of a minimal edge path through the anchor. Therefore, we *bind* them together, if λ_{b_1} does not differ much from λ_{b_2}, i.e. if $\frac{1}{\tau_2} < \frac{\lambda_{b_1}}{\lambda_{b_2}} < \tau_2$ for a threshold $\tau_2 \geq 1$, we add potentials

$$\beta \cdot \left(Rep(x_{b_1}, x_{b_2} \mid \lambda_2^4 \cdot (\lambda_{b_1}^4 + \lambda_{b_2}^4)/2) + Attr(x_{b_1}, x_{b_2})\right),$$

where λ_2 is a stretch factor for the length of binding edges, and β controls the importance of binding relative to the other potentials.

In summary, the objective function is made of nothing but attracting and repelling potentials that define a graph layout problem in the following way: Stations correspond to vertices with fixed positions, while Bézier points correspond to vertices to be positioned. Edges of different length exist between Bézier points and their anchors, between partners, and between Bézier points bound together. Just like edge lengths, repulsion differs across the elements. See Fig. 3 and recall that repelling forces act only locally (inside of neighborhoods). Let $\theta = (\varrho_1, \varrho_2, \lambda_1, \lambda_2, \beta, \tau_1, \tau_2)$ denote the vector of parameters. The effects of its components are summarized and demonstrated in Section 4.

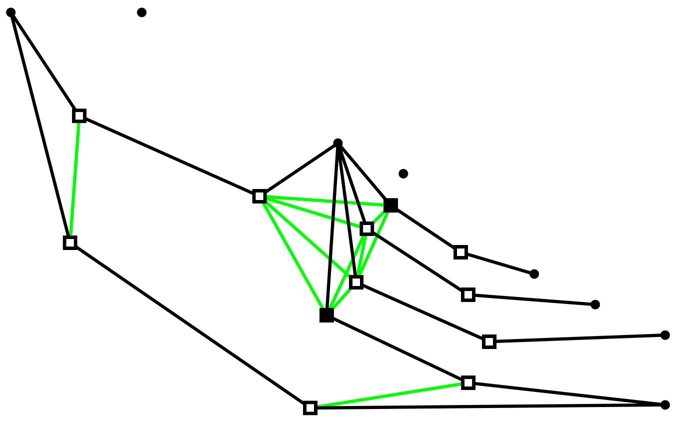


Fig. 3. Graph model of Bézier point layout dependencies for the train graph of Fig. 1(b). Note that there is no binding between the two layout elements indicated by black rectangles, because their distances from the anchor differ too much (threshold parameter τ_2)

4 Experiments

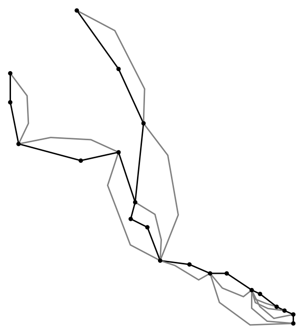
In order to obtain an objective function, we experimented with different potentials and parameters. We started with a simple combination of repelling forces from stations and attracting and repelling forces from partners and anchors. In fact, we first used splines to represent transitive edges. It seemed that they offered better control, since they actually pass through their control points. However, segments between partners tended to extend far into the layout area. After replacing splines by Bézier curves, the promising results encouraged us to try more elaborate objective functions. They showed that it is useful to represent long transitive edges straight-line, which led to the introduction of threshold τ_1. A new requirement we found after looking at earlier examples was that incident (consecutive or nested) transitive edges should lie on one side of a path of minimal edges. Binding proved to achieve this goal, but needed to be constrained to control segments of similar desired length using a threshold τ_2. Otherwise, short transitive edges are deformed when bound to a long one.

For convenience, we use the final combination of potentials and different choices of $\theta = (\varrho_1, \varrho_2, \lambda_1, \lambda_2, \beta, \tau_1, \tau_2)$ to demonstrate the effect of single parameters in Fig. 4. In particular, Fig. 4(d) shows why binding is a valuable refinement. The following table summarizes these effects:

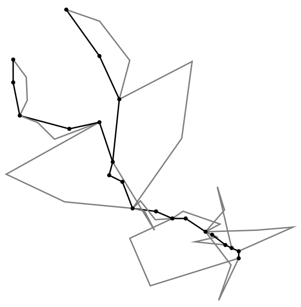
	controls
ϱ_1	distance of Bézier points from stations
ϱ_2	mutual distance of Bézier points
λ_1	length of control segments
λ_2	length of bands
β	importance of binding
τ_1	threshold for straight transitive edges
τ_2	threshold for binding segments of different length
ϵ_1	major axis radius of neighborhood defining ellipse
ϵ_2	minor axis radius of neighborhood defining ellipse

Next to a choice that proved appropriate (Fig. 4(a)), it is clearly seen how increased repelling forces spread Bézier points (Figs. 4(b) and 4(c)). Without binding, curves tend to lie on different sides of minimal edges (Fig. 4(d)). This can even be enforced (Fig. 4(e)).

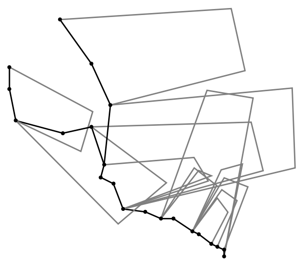
The identification of a suitable set of parameters is a serious problem. Mendonça and Eades use two nested simulated annealing computations to identify parameters of a spring embedder variant [9]. In [8], a genetic algorithm is used to breed a suitable objective function. However, both methods are heuristic in defining their objective as well as in optimizing it. Given one or more examples which are considered to be well done (e.g. by manual rearrangement), a theoretically sound approach would be to carry out parameter estimation for random variable $X(\theta)$ describing the layout model as a function of parameter vector θ.



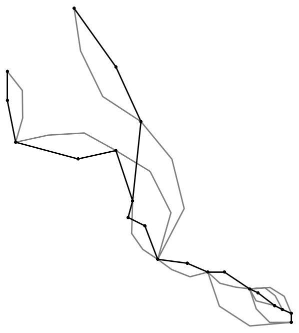
(a) Regular
$\theta = (0.3, 0.7, 0.7, 0.5, 0.4, 100, 2.2)$



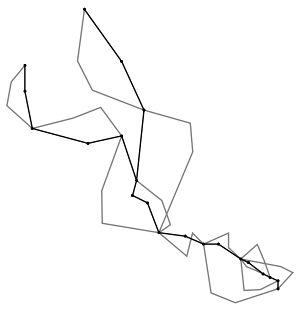
(b) Station repulsion
$\theta = (5, 0.7, 0.7, 0.5, 0.4, 100, 3)$



(c) Segment stretching
$\theta = (0.3, 4, 1, 0.5, 0.4, 100, 3)$



(d) No binding
$\theta = (0.3, 0.7, 0.7, 0, 0, 100, 0)$



(e) Inverse binding
$\theta = (0.3, 0.7, 0.7, 2, 1, 100, 3)$

Fig. 4. Effects of single parameters. For a better comparison, control segments are shown instead of the corresponding Bézier curves. All examples have $\epsilon_1 = 1.1$ and $\epsilon_2 = 0.5$

Given a layout x, the likelihood of θ is

$$P(X = x \mid \theta) = \frac{1}{Z(\theta)} \exp\{-U(x \mid \theta)\}$$

where $Z(\theta) = \sum_{y \in \mathcal{X}} \exp\{-U(y \mid \theta)\}$ is the normalizing constant. A maximum likelihood estimate θ^* is obtained by maximizing the above expression with respect to θ. Unfortunately, computation of $Z(\theta)$ is practically intractable, since it sums over all possible layouts. One might hope to reduce computational demand by exploiting the locality of random fields (see e.g. [11]). Even though neighboring layout elements are clearly not independent, reasonable estimates are obtained from the pseudo-likelihood function [1]

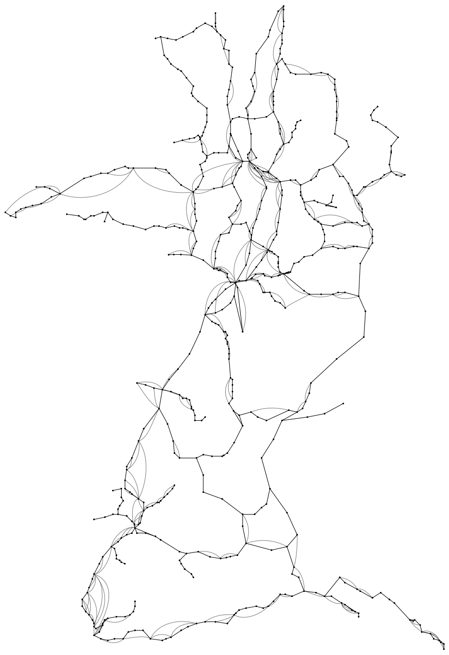
$$\prod_{l \in L} \frac{1}{Z_l(\theta)} \exp\left\{ - \sum_{C \in \mathcal{C}\colon l \in C} U_C(x \mid \theta) \right\}$$

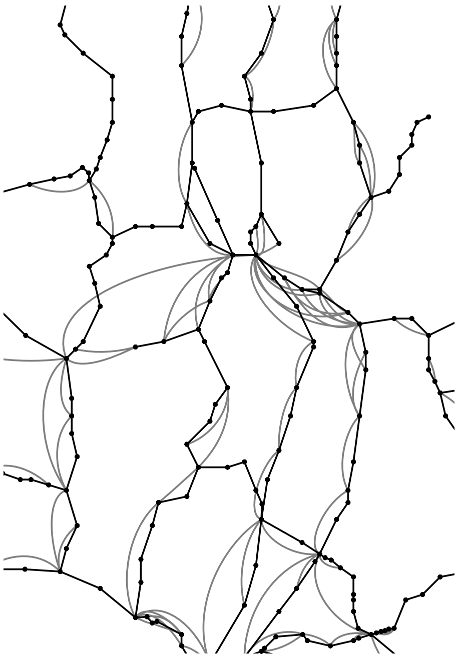
with $Z_l(\theta) = \sum_{y_l \in \mathcal{X}_l} \exp\{-\sum_{C \in \mathcal{C}: l \in C} U_C(\hat{x} \mid \theta)\}$, where $\hat{x}$ equals x with x_l replaced by y_l. However, $Z_l(\theta)$ is a sum over all possible positions of layout element l, such that maximization is still intractable in this setting. So we exploited locality in a very different way, namely by experimenting with small examples as in Fig. 4. The parameters θ thus identified proved appropriate, because the model scales so well.

To carry out the above experiments, and to generate large examples, we used an implementation of a fairly general random field layout module. It contains a set of fundamental neighborhood types and interaction potentials, to which others can be added. Since current concern is on flexibility and model design, a simple simulated annealing approach is used for energy minimization. Clearly, faster and more stable methods can be employed just as well. The original datasets provided by *TLC/EVA* are quite large. A train graph of roughly 2,000 vertices and 4,000 edges, for instance, is built from about 11 MByte of time table data. Connections are then classified into minimal and transitive edges. Existing code was used for these purposes.

The first example is shown in Fig. 5. The graph contains regional trains in south-west Germany. Edge classification, transformation into a layout graph, neighborhood generation, and layout computation took less than two minutes. Figs. 5(b) and 5(c) show magnified parts of the drawing from Fig. 5(a). Verify that connections can be told apart quite well, and that binding causes incident (consecutive or nested) transitive edges to lie on the same side of minimal edges.

Larger examples are given in Figs. 6 and 7. Computation times were about 35 minutes and 90 minutes, respectively, most of which was spent on generating the graph layout model and determining neighborhoods. One readily observes that the algorithm scales very well, i.e. increased size of the graph does not reduce layout quality on more detailed levels. This is largely due to the fact that neighborhoods remain fairly local. Together with the ability to zoom into different regions, data exploration is well supported. The benefits of a length threshold for curved transitive edges is another straightforward observation, notably in Fig. 7(a).


(a) Baden-Württemberg


(b) Ludwigshafen/Mannheim

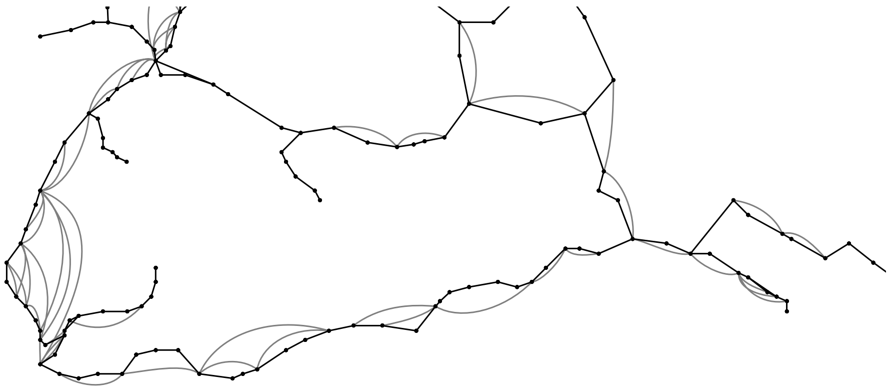
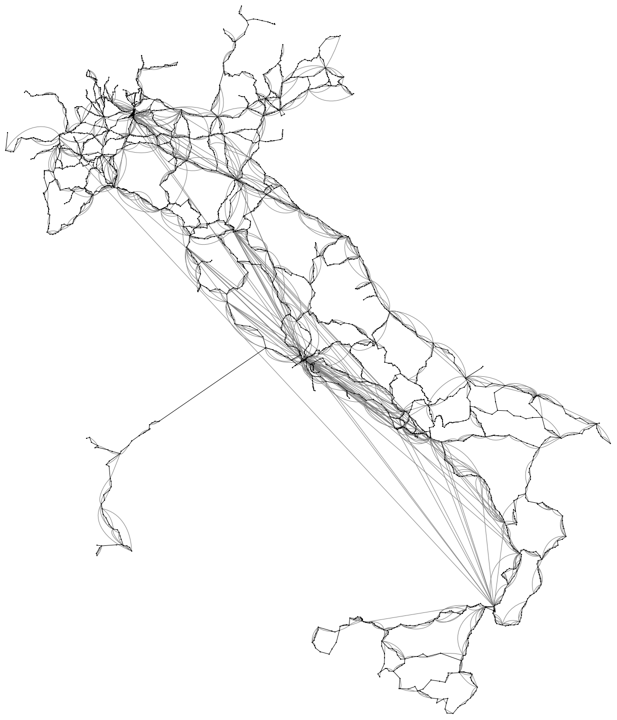

(c) Rhine from Konstanz to Basel to Freiburg

Fig. 5. Regional trains in south-west Germany: ca. 650 vertices, 900 edges (200 transitive), $\theta = (0.7, 0.3, 0.7, 0.5, 0.4, 100, 3)$



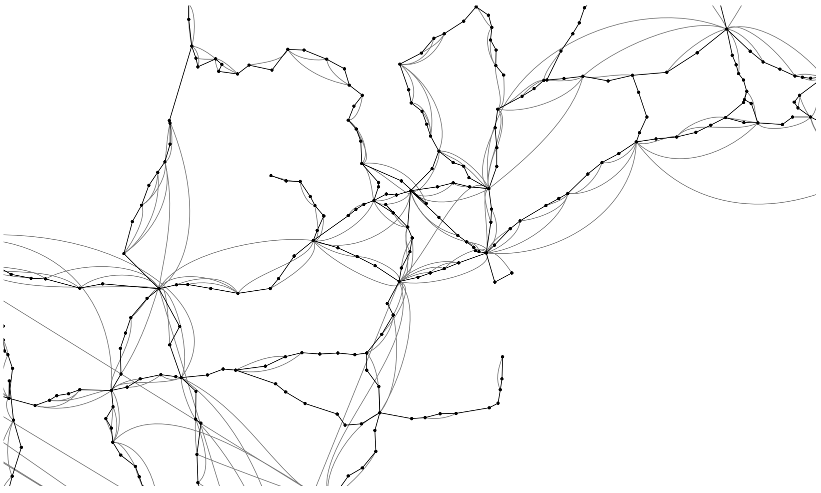
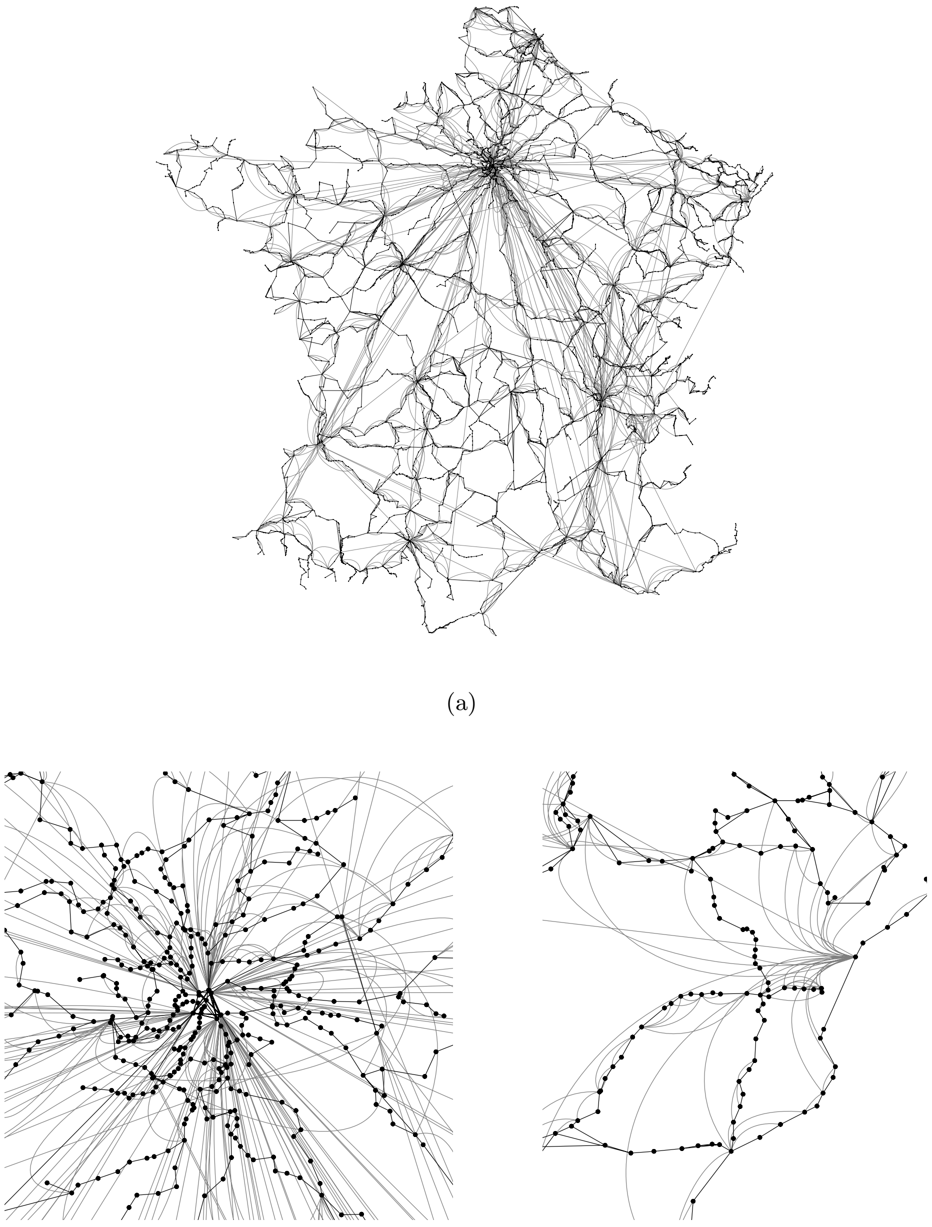


Fig. 6. Italian train and ferry connections: ca. 2,400 vertices, 4,400 edges (1,800 transitive), $\theta = (0.7, 0.3, 0.7, 0.5, 0.4, 100, 3)$. Zoom is into the surroundings of Venice



(a)

(b) Paris (note the long-distance stations!)

(c) Strasbourg

Fig. 7. French connections: ca. 4,500 vertices, 7,800 edges (2,500 transitive), $\theta = (0.7, 0.3, 0.7, 0.5, 0.4, 100, 3)$

Acknowledgments

We would like to thank Annegret Liebers, Karsten Weihe, and Thomas Willhalm for making the train graph generation and edge classification code available. Many thanks are due to Frank Müller, who implemented the transformation into a graph model, and to Vanessa Kääb, who implemented the current version of our general random field layout module.

References

[1] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, 48(3):259–302, 1986.

[2] Pierre Bézier. *Numerical Control.* John Wiley, 1972.

[3] Ulrik Brandes, Patrick Kenis, Jörg Raab, Volker Schneider, and Dorothea Wagner. Explorations into the visualization of policy networks. To appear in *Journal of Theoretical Politics.*

[4] Ulrik Brandes and Dorothea Wagner. A Bayesian paradigm for dynamic graph layout. *Proceedings of Graph Drawing '97.* Springer, Lecture Notes in Computer Science, vol. 1353, pages 236–247, 1997.

[5] Ulrik Brandes and Dorothea Wagner. Random field models for graph layout. Konstanzer Schriften in Mathematik und Informatik 33, University of Konstanz, 1997.

[6] Ron Davidson and David Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331, 1996.

[7] Peter Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[8] Toshiyuki Masui. Evolutionary learning of graph layout constraints from examples. *Proceedings of the ACM Symposium on User Interface Software and Technology.* ACM Press, pages 103–108, 1994.

[9] Xavier Mendonça and Peter Eades. Learning aesthetics for visualization. *Anais do XX Seminário Integrado de Software e Hardware*, Florianópolis, Brazil, pages 76–88, 1993.

[10] Marcello Pelillo and Edwin R. Hancock (eds.). *Energy Minimization Methods in Computer Vision and Pattern Recognition*, Springer, Lecture Notes in Computer Science, vol. 1223, 1997.

[11] Gerhard Winkler. *Image Analysis, Random Fields and Dynamic Monte Carlo Methods*, vol. 27 of *Applications of Mathematics.* Springer, 1995.