Journal of Graph Algorithms and Applications
http://www.cs.brown.edu/publications/jgaa/
vol. 4, no. 3, pp. 135–155 (2000)

Using Graph Layout to Visualize Train Interconnection Data

Ulrik Brandes *Dorothea Wagner*
Department of Computer & Information Science
University of Konstanz
http://www.inf.uni-konstanz.de/~{brandes,wagner}
{Ulrik.Brandes,Dorothea.Wagner}@uni-konstanz.de

Abstract

We consider the problem of visualizing interconnections in railway systems. Given time tables from systems with thousands of trains, we are to visualize basic properties of the connection structure represented in a so-called train graph. 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.

Positions of vertices in a train graph visualization are given by the geographical location of the corresponding station. If all edges are represented by straight-lines, the result is visual clutter with many overlaps and small angles between pairs of lines. We therefore present a non-uniform approach using different representations for edges of distinct meaning in the exploration of the data. Some edges are represented by curved lines, such that the layout problem consists of placing control points for these curves. We transform it into a graph layout problem and exploit the generality of the random field layout model formulation for its solution.

Communicated by G. Liotta and S. H. Whitesides: submitted November 1998, revised October 1999.

1 Introduction

The present layout problem 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. These comprise mostly train schedules; however, the data may also contain bus, ferry and even some pedestrian 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. Our aim is to aid visual inspection of this data, which is carried out at *TLC* to identify structural characteristics of (sub)networks and to back-up design decisions on extensions or modifications of networks. Reported future use will include evaluation support of schedules and pricing.

Figure 1 shows the kind of data that is provided. Since for even a moderately sized stop like the German part of the Konstanz main station there are about 100 trains regularly arriving or leaving, realistic input is quite large. To condense the input, a so-called *train graph* is built in the following way. For each regular stop of any train, a vertex is inserted into the graph. Two vertices are connected by exactly one edge if there is a point-to-point connection, i.e. some train runs from from one station to the other (or vice versa) without intermediate stops. Hence, the graphs considered here are simple and undirected.

An important part of the analysis is the classification of edges into 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 serving minor stations. On the other hand, transitive edges correspond to connections passing through other stations without halting. These are induced by through-trains. The information contained in a train graph is therefore the existence or absence of a point-to-point connection between pairs of stations, and the classification of each connection into minimal or transitive. Graphical presentation of the train graph and an edge classification computed in the analysis is desirable.

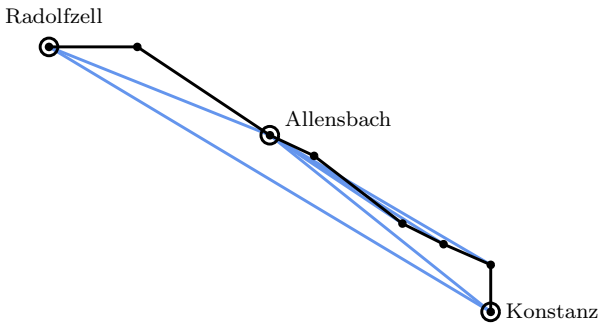
An edge classification is easily coded using color. Figure 2(a) shows a small part of a train graph with edges colored according to a precomputed classification. Stations are positioned according to their geographical location, and all edges are drawn as straight lines. Obvious graphical problems are edge overlaps and small angles between edges.

In order to maintain geographic familiarity, we are not allowed to move vertices, and minimal edges are best depicted by straight-lines, because they usually represent actual railways and should therefore not be the cause of the problem. It seems therefore reasonable to change the representation of transitive edges to curves, as depicted in Figure 2(b). They provide the flexibility to route an edge such that overlaps and small angles are resolved. In general, representation of non-stop connections by curved lines not only helps to reduce visual clutter and ambiguity, but also directly resembles the intuition of fast

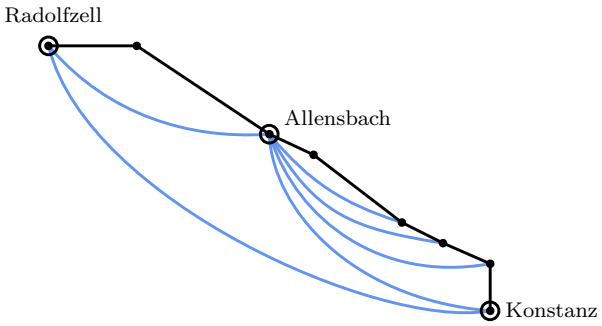
```
*Z 05130 85    01
*G SE  8506131 8001790
*A VE 8506131 8001790 000000
*A G  8506131 8001790
8506131 Kreuzlingen              1112  (...)
8003400 Konstanz            1115 1125  (...)
8003401 Konstanz-Petersh.   1127 1128  (...)
8003416 Konstanz-Wollmat    1130 1130  (...)
8004997 Reichenau(Baden)    1132 1133  (...)
8002683 Hegne               1135 1135  (...)
8000496 Allensbach          1138 1138  (...)
8003872 Markelfingen        1143 1143  (...)
8000880 Radolfzell          1147 1149  (...)
8001059 Böhringen-Rickelsh. 1152 1152  (...)
8000073 Singen(Hohentwiel)  1158 1200  (...)
8004107 Mühlhausen(b Engen) 1206 1206  (...)
8006321 Welschingen-Neuhaus. 1209 1209 (...)
8001790 Engen               1212       (...)
```

```
(...)
8000880 Radolfzell            -58.5  -510.8   (...)
(...)
8003400 Konstanz              -43.5  -519.8   (...)
8003401 Konstanz-Petersh.     -43.5  -518.2   (...)
8003416 Konstanz-Wollmat      -45.1  -517.5   (...)
(...)
8506131 Kreuzlingen           -40.2  -524.5   (...)
(...)
```

Figure 1: Schedule of a single train and excerpts from a station list. The schedule lists all stations used by the train with arrival and departure times. Every station has a unique identification number, and coordinates are in kilometers relative to the city of Hannover (irrelevant data omitted)



(a) straight-line segments



(b) Bézier curves

Figure 2: Different representations of transitive edges in a small train graph

vehicles passing by minor stops.

To render Bézier curves, control points need to be positioned. Using the framework of random field layout models introduced in [3], 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 currently used to identify suitable layout models for social and policy networks [4, 3]. These 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 Section 2, we review briefly the concept of random field layout models. A specific random field model for train graph

layout is defined in Section 3. Section 4 features a short discussion on aspects of parametrization and experiments with real-world examples.

2 Random Field Layout Models

In this section we review briefly the uniform graph layout formalism introduced in [3]. As can be seen from Section 3, model prototyping 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 representations, for instance, are completely determined by an assignment of coordinates to each vertex. However, straight-line representations are but one special case of a layout problem. In the most general formulation, each element of a set $L = \{l_1, \ldots, l_k\}$ of arbitrary *layout elements* is assigned a value from a set of feasible 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 need 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 $U : \mathcal{X} \to \mathbb{R}$ is defined. Since it is difficult to judge the quality of a layout as a whole, the objective function evaluates configurations of small 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* $\eta = \{\eta_l \,|\, l \in L\}$, 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. There is an edge in E^η between two layout elements, if one is in the neighborhood of the other. The interactions are symmetric by definition, i.e. we require $l_2 \in \eta_{l_1} \Leftrightarrow l_1 \in \eta_{l_2}$ for all $l_1, l_2 \in L$, so that 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 $U_C : \mathcal{X} \to \mathbb{R}$ 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 $U : \mathcal{X} \to \mathbb{R}$ 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, and can be interpreted as the amount of distortion in the layout.

Fundamental potentials. One advantage of separating the energy function into interaction potentials of small subsets of layout elements is that recurrent design principles can be isolated to form a toolbox of fundamental criteria. Not

surprisingly, two central potentials are those corresponding to the forces used in the spring embedder [7]:[1]

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

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

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

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

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

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

- *Distance Potential:* Since $\mathsf{Rep}(x_k, x_l \,|\, \lambda^4) + \mathsf{Attr}(x_k, x_l \,|\, 1)$ is minimized when $d(x_k, x_l) = \lambda$, one can specify a desired distance between two layout elements (e.g. edge length) by

$$U^{(\mathsf{dist})}_{\{k,l\}}(x) = \mathsf{Dist}(x_k, x_l) = \mathsf{Rep}(x_k, x_l \,|\, \lambda^4) + \mathsf{Attr}(x_k, x_l \,|\, 1)$$

  where $\mathsf{Dist}(x_k, x_l \,|\, \lambda^4)$ is used like above.

Note that many other design rules (sufficiently large angles, vertex-edge distance, edge crossings, etc.) are easily formulated in terms of interaction potentials [3].

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 [5]). See [13] 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. [12]).

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

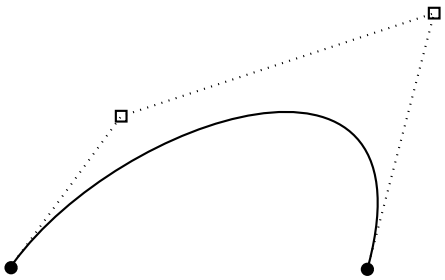


Figure 3: Bézier cubic curve [2]. Two endpoints and two control points define a smooth curve that is entirely enclosed by the convex hull of these four points

3 A Layout Model for Curved Edges

We now define a layout model for undirected train graphs $G = (V, E)$. The layout elements that need to be positioned to render Bézier curves are their control points. In fact, we may consider stations and control points to be vertices of an auxiliary graph, so that rules for favorable positioning can be modeled by auxiliary edges of appropriate desired length.

Their geographical location gives the position of all vertices corresponding to stations, and we identify these vertices with their position. Minimal edges as well as very long transitive edges are represented straight-line. For the other edges we use Bézier cubic curves (cf. Figure 3).[2] Let $\breve{E}_{\tau_1} \subseteq E$ 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 $\breve{E}_{\tau_1}$, $L = \left\{ b_u(e), b_v(e) \,|\, e = \{u, v\} \in \breve{E}_{\tau_1} \right\}$. If two Bézier points belong to the same edge, they are called *partners*. The *anchor*, $a_{b_v(e)}$, of any $b_v(e) \in L$ 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.

The position assigned to a Bézier point is influenced by its partner, its anchor, all Bézier points with the same anchor or close default positions, and all stations near the default position. Let $\{u, v\} \in \breve{E}_{\tau_1}$ 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 and its anchor, 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 are symmetric) the set of Bézier points b' for which $b \in \mathcal{E}_{b'}$. We used $\epsilon_1 = 1.1$ and $\epsilon_2 = 0.5$ for the examples presented in Section 4.

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

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 $\{u, v\} \in \breve{E}_{\tau_1}$, there are repulsion potentials

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

  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 repulsion and attraction potential

$$\mathsf{Dist}(x_b, a_b \mid (\lambda_1 \cdot \lambda_b)^4)$$

  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

$$\mathsf{Rep}(x_{b_1}, x_{b_2} \mid \varrho_2^4 \cdot \min\{\lambda_{b_1}^4, \lambda_{b_2}^4\})$$

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

$$\mathsf{Dist}(x_{b_1}, x_{b_2} \mid (\lambda_1 \cdot \lambda_{b_1})^4)$$

- *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

$$\beta \cdot \mathsf{Dist}(x_{b_1}, x_{b_2} \mid \lambda_2^4 \cdot (\lambda_{b_1}^4 + \lambda_{b_2}^4)/2)$$

  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 attraction and repulsion potentials that define an auxiliary 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 desired lengths exist between Bézier points and their anchors, between partners, and between Bézier points bound together. Just like edge lengths, the magnitude of repulsion differs across the elements. See Figure 4 and recall that repulsion potentials are defined on local neighborhoods only. The respective influence of the different parameters is discussed in the following section.

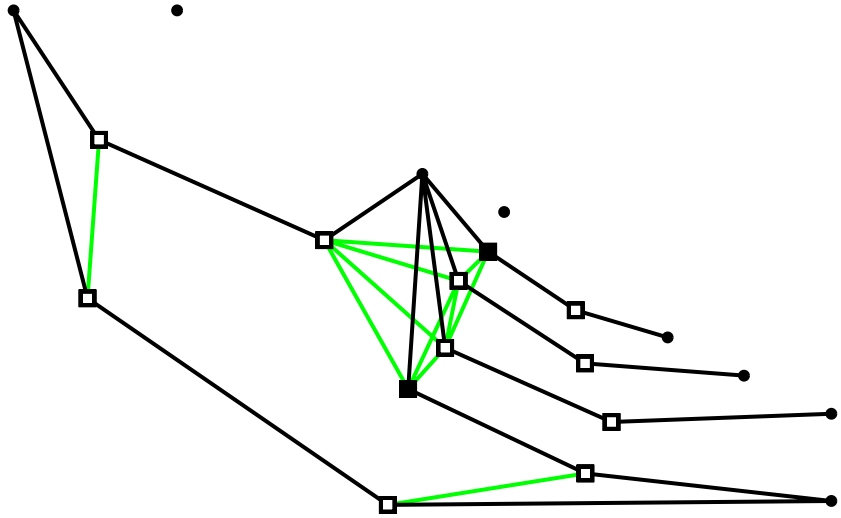


Figure 4: Auxiliary graph induced by Bézier point layout interactions for the train graph of Figure 2(b). Note that there is no binding between the two layout elements indicated by black rectangles, because their default distances from the anchor differ too much (threshold parameter τ_2)

4 Experiments

The objective function described in the previous section was obtained only after experimentation with a number of different potentials and parameters. We started with a simple combination of repulsion from stations and attraction and repulsion from partners and anchors. In fact, we then used splines to represent transitive edges. It seemed that they offered better control, since they actually pass through their control points. However, spline 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. In particular it 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 while discussing earlier examples with users 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, because otherwise short transitive edges are deformed when bound to long ones. Threshold τ_2 therefore controls the length ratio of segments bound.

Identification of a suitable vector $\theta = (\varrho_1, \varrho_2, \lambda_1, \lambda_2, \beta, \tau_1, \tau_2)$ of parameters is a serious problem. Two nested simulated annealing computations are used in [11] to identify parameters of a spring embedder variant. In [9], a genetic algorithm is used to breed a suitable objective function. However, both meth-

ods 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. Given a layout x, the *likelihood* of θ is

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

where $Z(\theta) = \sum_{y \in \mathcal{X}} \exp\{-U(y \,|\, \theta)\}$ 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. [13]). Even though neighboring layout elements are clearly not independent, reasonable estimates are obtained from the *pseudo-likelihood* function [1]

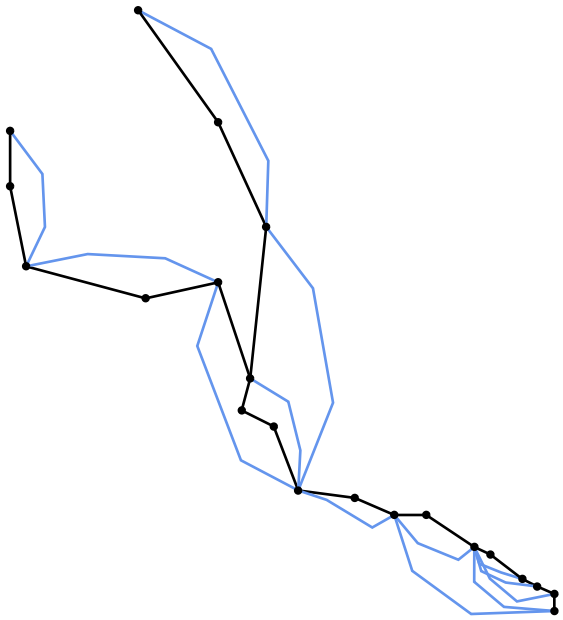
$$\prod_{l \in L} \frac{1}{Z_l(\theta)} \exp\left\{ - \sum_{C \in \mathcal{C} \,:\, l \in C} U_C(x \,|\, \theta) \right\}$$

with $Z_l(\theta) = \sum_{x_l \in \mathcal{X}_l} \exp\{-\sum_{C \in \mathcal{C} \,:\, l \in C} U_C(x \,|\, \theta)\}$. 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 exploit locality in a very different way, namely by experimenting with small examples in a feedback cycle. The parameters θ thus identified prove appropriate even for huge graphs, indicating that the local neighborhood definition lets the model scale well.

The rationale behind each component of $\theta = (\varrho_1, \varrho_2, \lambda_1, \lambda_2, \beta, \tau_1, \tau_2)$ is listed in Figure 5, as well as a choice of values that proved satisfactory. The effects of some parameters are demonstrated in Figure 6. It is clearly seen how increased repulsion potentials spread Bézier points (Figs. 6(a) and 6(b)). Without binding, curves tend to lie on different sides of minimal edges (Fig. 6(c)), which can even be enforced (Fig. 6(d)). This indicates why binding is a valuable refinement.

To carry out the above experiments and to generate large examples, we initially used an implementation of a fairly general random field layout module, written in C++ using LEDA [10]. It provides a set of fundamental neighborhood types and interaction potentials, to which others can be added. Since our main goals with this module are flexibility and model design, a simple simulated annealing approach is used for energy minimization. Since it turned out that the final model needed only attraction and repulsion potentials, we later replaced the module with a customized implementation of the approach of [8], which sped up energy minimization by a factor of ten. All running times given are with respect to this latter implementation executed on one 336 MHz Ultra-SPARC-II processor of a SUN Enterprise 4000/5000 running under Solaris 2.5.1 with 1024 MBytes of RAM. Note that neighborhoods are computed in a preprocessing step, and we have made no effort whatsoever to reduce its running time.

The original datasets provided by *TLC/EVA* are quite large: For a train graph of the size shown in Figure 10 (roughly 2,000 vertices and 4,000 edges),

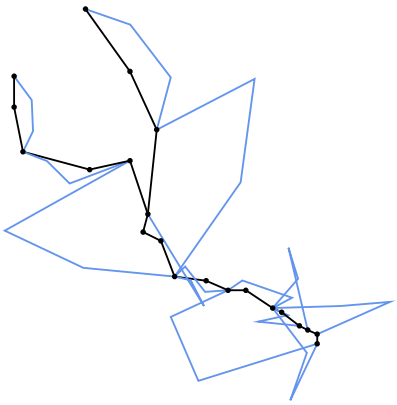


(a) Small part of a train graph with parameters $\theta = (0.3, 0.7, 0.7, 0.5, 0.4, 100, 2.2)$

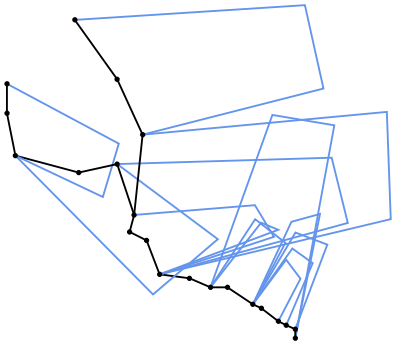
θ	controls
ϱ_1	distance of Bézier points from stations
ϱ_2	mutual distance of Bézier points
λ_1	length of control segments
λ_2	length of bands
β	importance of binding
τ_1	threshold for straight transitive edges
τ_2	threshold for binding segments of different length
ϵ_1	major axis radius of neighborhood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

(b) Parameters of the train graph layout model

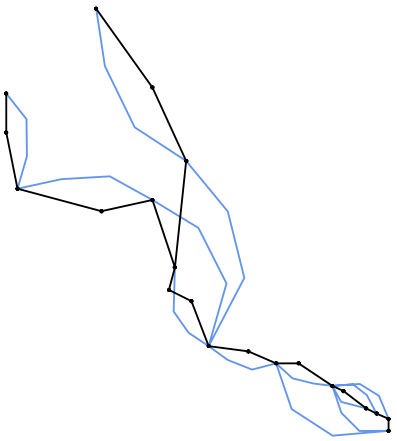
Figure 5: User specifiable parameters in the train graph layout model and a recommended choice applied to a small train graph. Control segments shown instead of Bézier curves (cf. Figure 6)



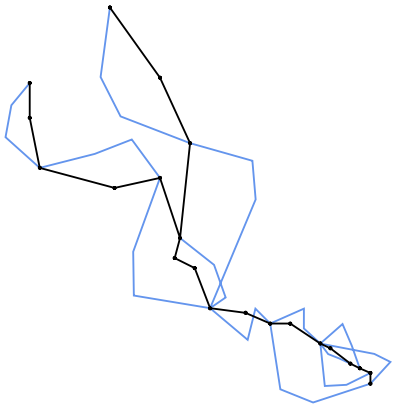
(b) Station repulsion
$\theta = (\boxed{5}, 0.7, 0.7, 0.5, 0.4, 100, 3)$



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



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



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

Figure 6: Effects of some parameters demonstrated. For ease of comparison, control segments are shown instead of the corresponding Bézier curves. All examples have $\epsilon_1 = 1.1$ and $\epsilon_2 = 0.5$ and should be compared to Figure 5

about 11 MBytes of time table data are evaluated. Connections are classified into minimal and transitive edges using existing code.

The first example is shown in Figure 7. The graph represents regional trains in southwest Germany. Edge classification, transformation into a layout graph, neighborhood generation, and layout computation took less than 10 seconds. The example also demonstrates how visual inspection can immediately yield some candidates for misclassified edges. Parts of the drawing are magnified in Figures 8 and 9. A few labels have been added to support geographical location of the area shown, but otherwise the drawings have not been modified. Note that connections can be told apart quite well, and that binding successfully causes incident (consecutive or nested) transitive edges to lie on the same side of minimal edges.

Larger examples are given in Figures 10 and 12. Computation times were about 5 minutes and 9 minutes, respectively, most of which was spent on determining the neighborhoods. Energy minimization took about 30 seconds and 47 seconds, respectively. 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 (Figs. 11 and 13). This is largely due to the fact that neighborhoods remain fairly local. The benefits of a length threshold for curved transitive edges is another straightforward observation, notably in Figures 12 and 13(a). Together with the ability to zoom into different regions, data exploration is well supported.

Acknowledgments

Besides our contacts at *TLC*, we would like to thank Annegret Liebers, Karsten Weihe, and Thomas Willhalm for making the train graph generation and edge classification code available. We are grateful to Frank Müller, Vaneesa Kääb, and Marco Gaertler, who carried out most of the other implementation work. We also wish to thank the referees for some helpful suggestions.

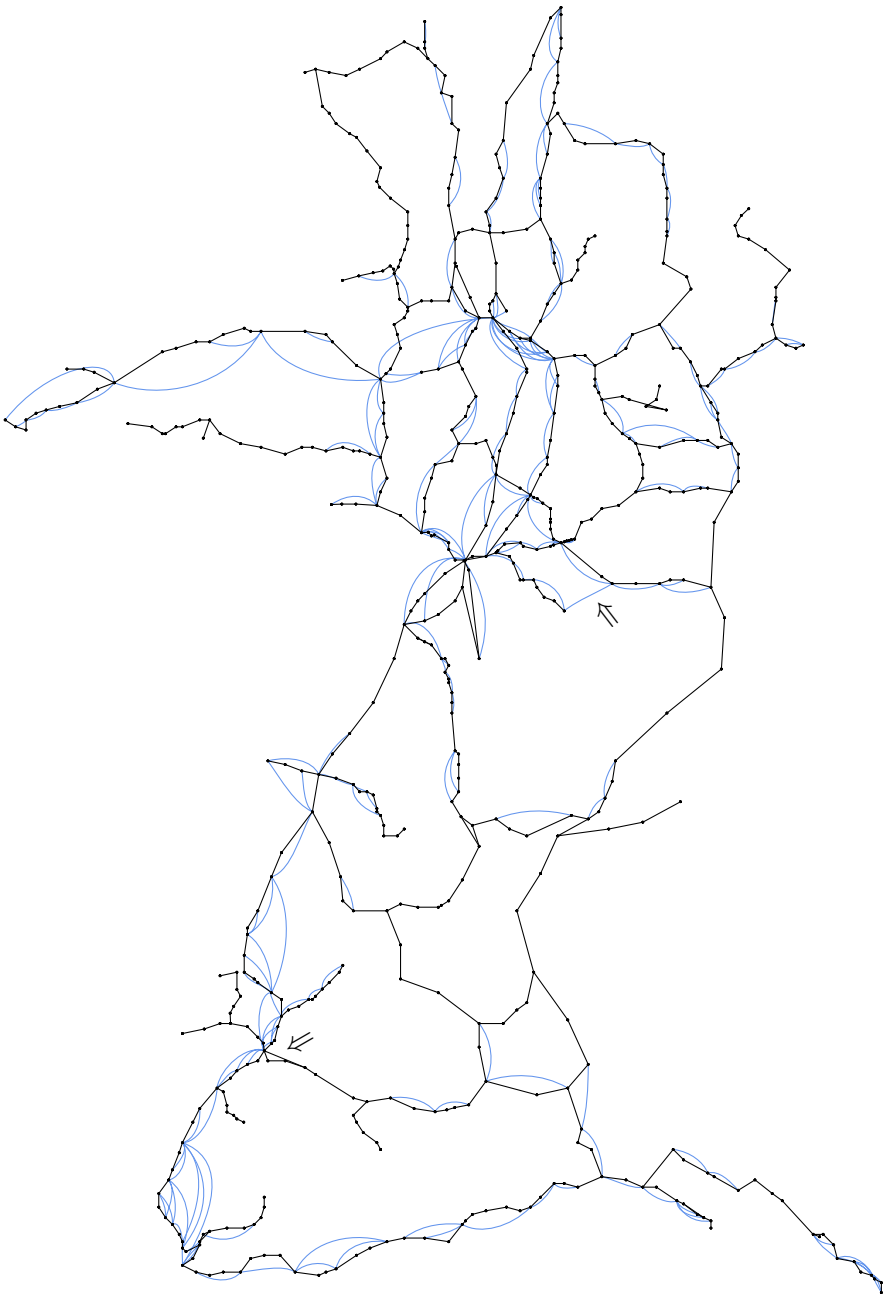


Figure 7: Regional trains in southwest Germany. 619 vertices, 876 edges (229 transitive), $\theta = (0.7, 0.3, 0.7, 0.5, 0.4, 100, 3)$. Arrows indicate two out of several edges that appear to be misclassified

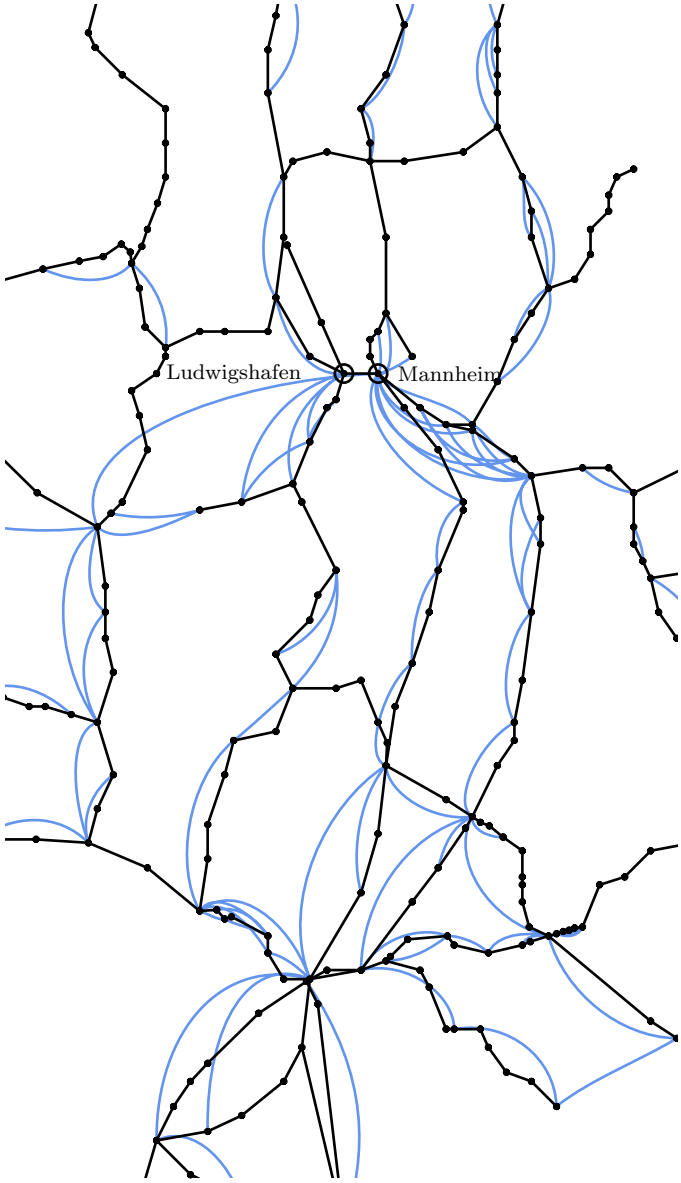


Figure 8: Magnification from Figure 7

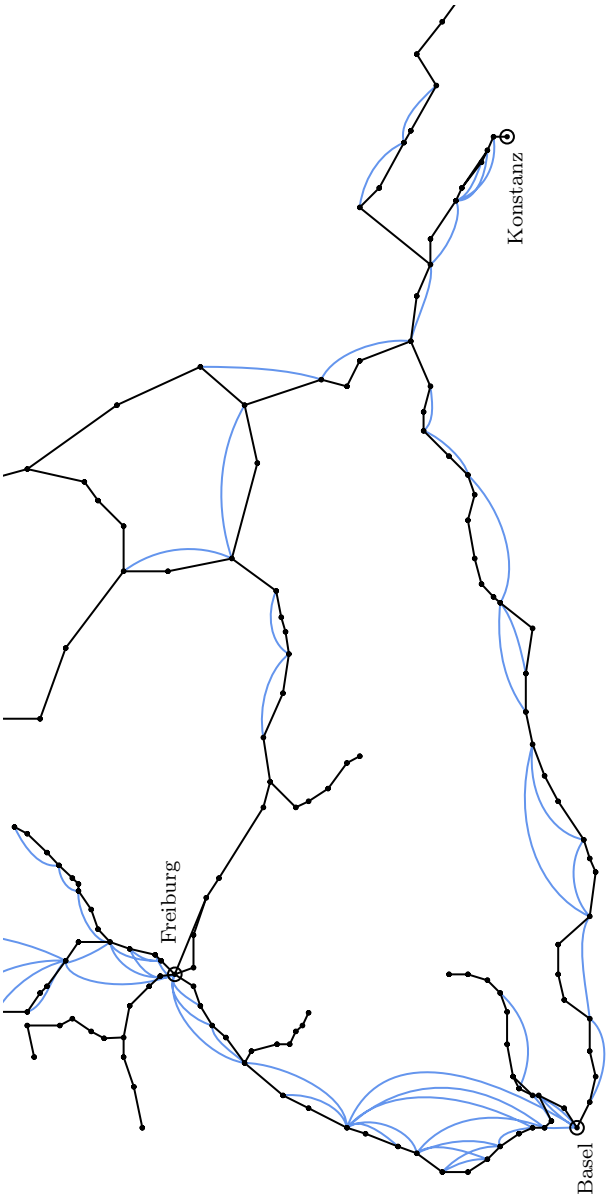


Figure 9: Magnification from Figure 7

Figure 10: Italian train and ferry connections. 2,386 vertices, 4,370 edges (1,849 transitive), $\theta = (0.7, 0.3, 0.7, 0.5, 0.4, 100, 3)$

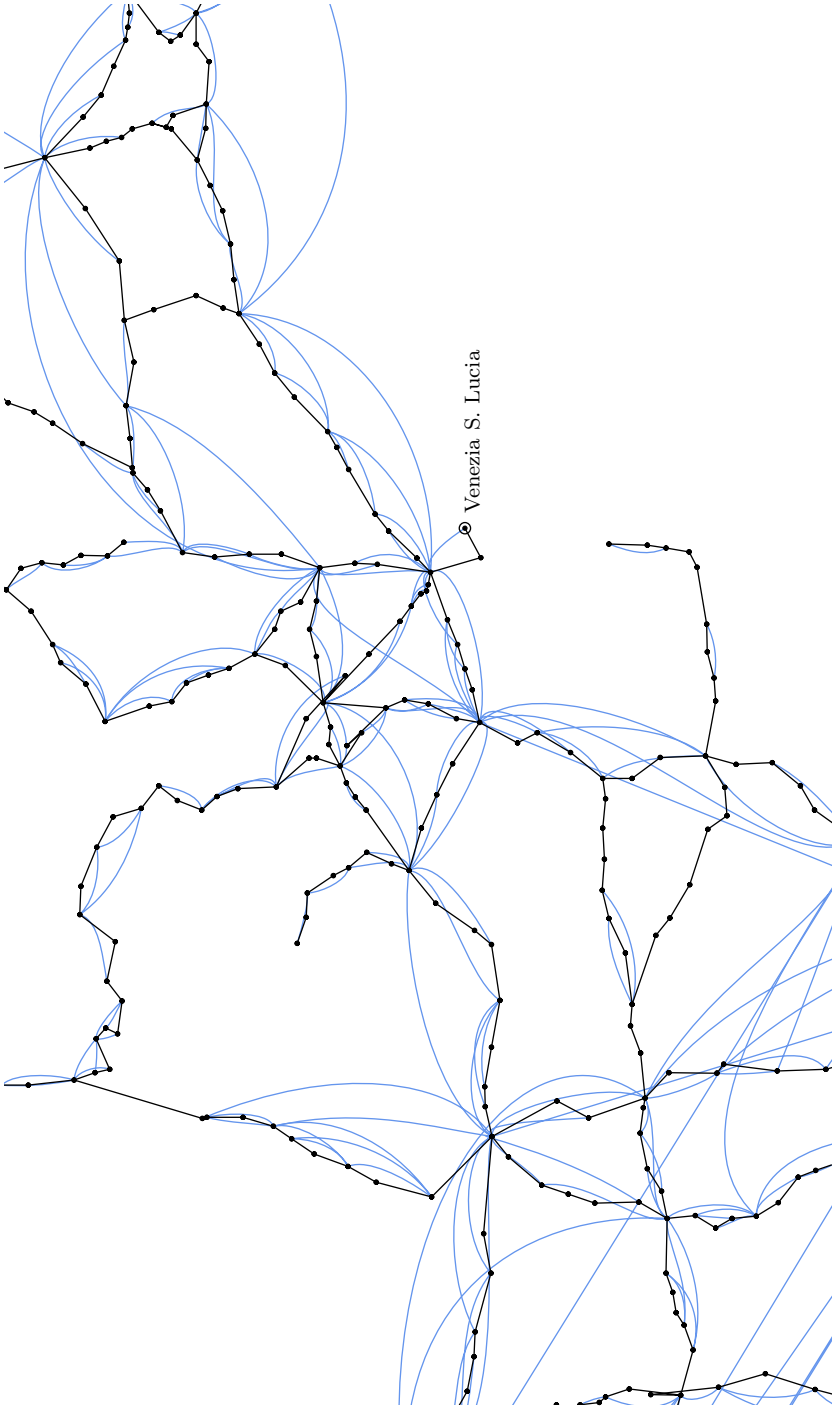


Figure 11: Magnification from Figure 10

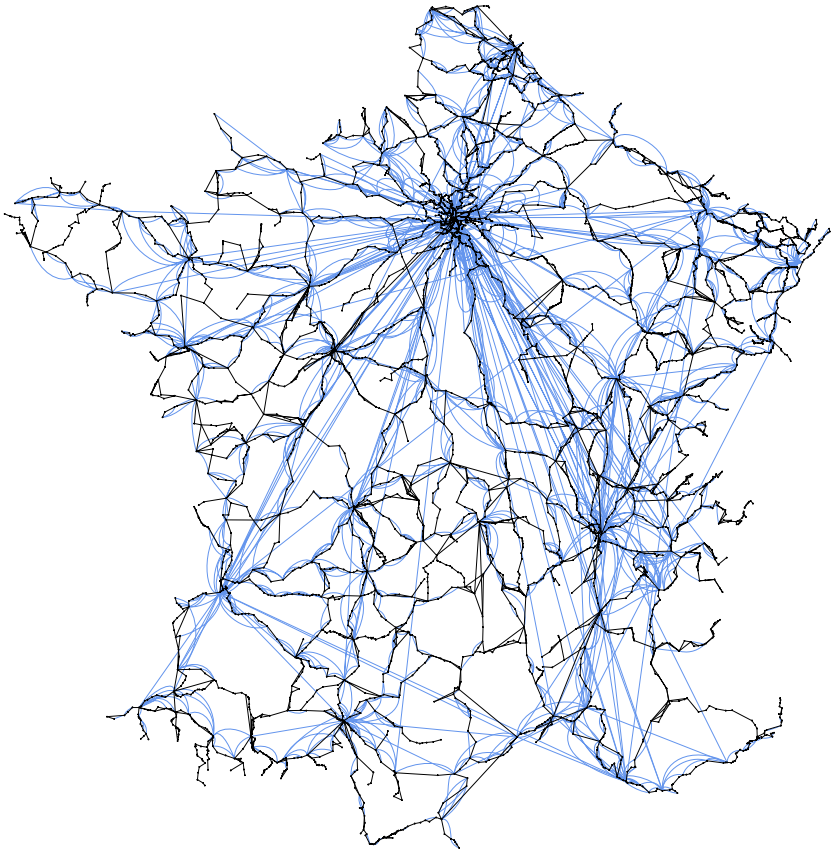
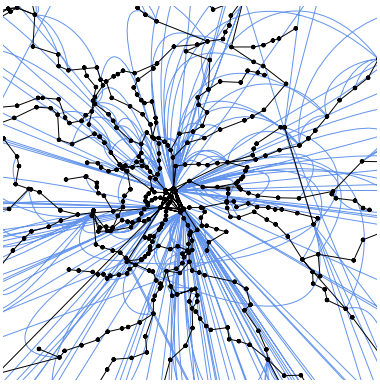
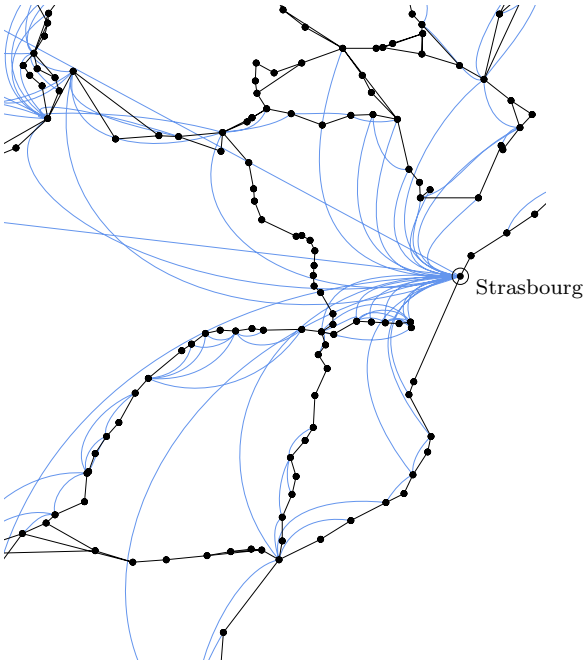


Figure 12: French connections. 4,551 vertices, 7,793 edges (2,408 transitive), $\theta = (0.7,\ 0.3, 0.7, 0.5, 0.4, 100, 3)$



(a) Paris has six long-distance stations



(b) Strasbourg, gateway to France

Figure 13: Magnifications from Figure 12

References

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

[2] P. Bézier. *Numerical Control*. Wiley, 1972.

[3] U. Brandes. *Layout of Graph Visualizations*. PhD thesis, University of Konstanz, 1999. See `http://www.ub.uni-konstanz/kops/volltexte/1999/255/`.

[4] U. Brandes, P. Kenis, J. Raab, V. Schneider, and D. Wagner. Explorations into the visualization of policy networks. *Journal of Theoretical Politics*, 11(1):75–106, 1999.

[5] U. Brandes and D. Wagner. A Bayesian paradigm for dynamic graph layout. In G. Di Battista, editor, *Proceedings of the 5th International Symposium on Graph Drawing (GD '97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 236–247. Springer, 1997.

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

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

[8] T. M. Fruchterman and E. M. Reingold. Graph-drawing by force-directed placement. *Software—Practice and Experience*, 21(11):1129–1164, 1991.

[9] T. Masui. Evolutionary learning of graph layout constraints from examples. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '94)*, pages 103–108. ACM, The Association for Computing Machinery, 1994.

[10] K. Mehlhorn and S. Näher. *The* LEDA *Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999. Project home page at `http://www.mpi-sb.mpg.de/LEDA/`.

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

[12] M. Pelillo, editor. *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR '97)*, volume 1223 of *Lecture Notes in Computer Science*. Springer, 1997.

[13] G. Winkler. *Image Analysis, Random Fields and Dynamic Monte Carlo Methods*, volume 27 of *Applications of Mathematics*. Springer, 1995.