Empirical Design of Geometric Algorithms

Karsten Weihe[1] Ulrik Brandes[1] Annegret Liebers[1] Matthias Müller–Hannemann[2]
Dorothea Wagner[1] Thomas Willhalm[1]

Abstract

The computer–aided solution to algorithmic problems is becoming more and more important in various application domains. This is in particular true for computational geometry. For example, geometric problems naturally arise in image processing, computer graphics, and all kinds of computer–aided design, just to mention a few. Even more, the general tendency towards the application of visual aids in virtually all fields of science, technology, and business raises many new, unexpected geometric challenges.

A sound mathematical treatment of these problems and a systematic computational study on the resulting algorithms are desirable. However, in practice, there are often obstacles to such an attempt. In this paper, we will systematically discuss our experiences with a few obstacles that occurred in four of our projects and significantly influenced our reasoning on algorithms in each of them.

1 Introduction

In applied algorithmic projects one is often confronted with obstacles that are usually not addressed in the literature on algorithms. In fact, most of these obstacles might be rather technical in nature and do not have any impact on the design and analysis of the algorithms. However, in various projects we were also faced with obstacles that are inherent in the respective real-world problems and had to be taken into account both in the theoretical and the empiricial algorithmic research.

This paper is about these obstacles and about four of our projects, in which obstacles of this kind occurred in a geometric setting. These projects stem from different application domains (traffic logistics, computer–aided design and numerical analysis, sociology) and address geometric problems that do not seem to be related in any way. However, in retrospect, it has turned out that the obstacles and the strategies to tackle them were quite similar. It seems to us that this coincidence is not by chance, so a general, comparative, retrospective analysis of finished and on-going projects may be worth the effort.

More specifically, we will restrict the discussion to the following obstacles, which occurred in various ways:

- *Formalization Obstacle*: The algorithmic problem is too complex to understand all relevant details and their interrelations. Obvious reductions to intelligible formulations seem to miss crucial aspects. In the worst case, the outputs of an algorithm that is based on such a reduced problem definition may be useless or even misleading in view of the real problem.

  Moreover, the algorithmic problem may involve subjective aspects. These aspects may be at the heart of the matter and thus must not be abstracted away.

- *Maintenance/Volatility Obstacle*: Good performance in terms of run time and space consumption is always desirable. However, further criteria may require a compromise. For example, maintenance cost is immediately affected by the "smartness" of an algorithm: the more sophisticated an algorithm is, the more difficult it is to maintain (especially since maintenance is usually done by software developers who are not experts in computational geometry).

  Maintenance may be infeasibly expensive if the "logic" of an algorithm is affected. This may indeed happen in practice because the specification of the algorithmic problem may depend on context-specific, "volatile" aspects of the application domain.

  High maintenance cost of a piece of software may result in a very short life time [15], which means that, after all, the project was not really a success.

- *Data Obstacle*: Typically, real–world data are not worst—case data, nor do they necessarily come close to the average-case data produced by obvious artificial (*e.g.* random) instance generators. The intrinsic properties that determine the tractability of real–world data are not always sufficiently understood to formalize them. Nonetheless, it may be necessary to make use of these properties in order to design an appropriate algorithm.[3]

[1] Universität Konstanz, Informatik, Fach D188, 78457 Konstanz, Germany, *last–name*@fmi.uni-konstanz.de, www.fmi.uni-konstanz.de/~*last–name*/

[2] Technische Universität Berlin, Mathematik Sekr. MA 6–1, 10623 Berlin, Str. d. 17. Juni 136, Germany, mhannema@math.tu–berlin.de, http://www.math.tu–berlin.de/~mhannema/

[3] An extreme example was encountered in [18]: an application of the *hitting-set problem*, which is $\mathcal{NP}$-hard. The real-world instances were large, however, a few simple reduction techniques made all of them amenable to a straightforward brute-force approach. The crucial intrinsic properties of the data, which made this strategy a success, are by no means understood up to now.

Intent of the paper. We will discuss these obstacles in view of the four above-mentioned projects. Since these four projects are quite different by nature, we believe that a summary and analysis of common aspects may give interesting new insights (even beyond computational geometry).

Roughly speaking, the general idea is to replace the formal definition of the algorithmic problem by some kind of *framework* for the problem definition, in which various degrees of freedom (the "little screws") can be used to approximate the real problem *interactively* step-by-step. The idea is to make use of domain knowledge *beyond the degree to which we can formalize it.*

In fact, if the outputs are well presented graphically, a domain expert may at least compare two or more outputs "by eye" and give a profound opinion which output is preferable.[4] In turn, this information may be used to adjust the "screws" a little bit better and submit the result to the human observer again. The hope is that an "expert judgement" based on such a presentation suffices to adjust the screws appropriately in an iterative process. Our concrete experiences suggest that this hope is justified in practice.

These "screws" and a meaningful graphical presentation do not necessarily come for free; they must be treated as first-class algorithmic design goals like correctness and efficiency. Fortunately, there is often some space for a reasonable compromise. In fact, the rigorous mathematical meaning of *correctness* must anyway be relaxed in the presence of the *Formalization Obstacle*; on the other hand, a certain, suboptimal degree of *efficiency* is sufficient in many applications.

Methodology of the analysis. In Section 2, we will illustrate these obstacles by reflecting on the four above-mentioned projects. Afterwards, in Section 3, we will summarize the common conclusions in retrospect.

Throughout the last few years, people from the computational geometry community and related theoretical communities have been collecting a large body of experiences with various applications. For example, two invited talks given in the WAE '97 conference demonstrated this [9, 11].

These talks reflected on experiences gained from past work, and this is also our approach. In fact, in various fields of natural and engineering sciences, and even more in social sciences and liberal arts, reflection has a long–standing, established tradition as one of the main research methodologies. In case of interdisciplinary efforts, the traditions of all participating disciplines might be valuable for narrowing the gap between theory and practice.

[4]Of course, in an interdisciplinary project the algorithmician has to become a "quasi-expert" in the application domain anyway and may thus take over the role of the domain expert to some extent.

2 Projects

In the four subsections of this section, we will discuss the individual projects. Each of these subsections follows a common format: an *Introduction*, a discussion of the above-mentioned *Obstacles*, a description of an algorithmic *Framework* for the respective problem definition, and a description of an attempt for visual *Validation* based on implicit human domain knowledge.

2.1 Mesh Generation in CAD

Introduction. Mesh generation is an automatic step in the computer-aided design (CAD) of traffic vehicles, machines, engines, and the like. In our concrete case, we are faced with surfaces in the three-dimensional space, which are modeled as coarse meshes of non-plane polygons (so-called *free-form surfaces*). Some of them are surfaces of solid bodies; other surfaces approximate hollow bodies, which are regarded as two-dimensional objects in the three-dimensional space. See Figure 1 for an example of hollow bodies.

The algorithmic problem is to refine such a coarse mesh into a *conforming* all-quadrilateral mesh (Figure 2). In that, *conforming* means that adjacent quadrilaterals share a whole side (or a single vertex). Figure 1 demonstrates that input meshes are usually not conforming.

The purpose of this refinement is to prepare the mesh for a numerical analysis by means of the finite-element method. The objective is a refinement on which the finite-element method is efficient and provides a solution of high quality.[5] Roughly speaking, a mesh is suitable if it meets some specified density requirements, all quadrilaterals are approximately square-shaped, and the overall structure of the mesh looks very regular (like in Figure 2).[6]

Technical reasons may induce further side constraints. For example, the original problem formulation given to us included a small set of *templates*, and each polygon of the input mesh has to be refined according to one of them. Figure 3 shows typical templates to refine a quadrilateral into quadrilaterals only (except for part (e), which shows an "emergency template" with one isolated triangle).

Previous work on this problem either uses the template approach [16], advancing-front based heuristics like paving [3], or grid-based approaches [10]. To our knowledge, none of the algorithmic work published so far was based on a rigorous mathematical model that captures more than a few selected aspects of the problem.

A general survey on mesh generation can be found in [8], more recent overviews of this active research field are available on the Web.[7] A few related questions have been considered from a computational geometry viewpoint (see [1, 17] for a survey).

[5]Efficiency of the refinement algorithm itself (though not negligible) is not a major concern, since the overall run time of a CAD cycle is usually totally dominated by the finite-element method. This is an example of a point we mentioned in the Introduction ("Intent of the paper"): the practical efficiency requirements leave enough space to incorporate other criteria as well.

[6] A domain expert once told us as a general rule of thumb:
"a mesh is good if it looks beautiful."

[7]`http://www.andrew.cmu.edu/user/sowen/mesh.html` and `http://www.users.informatik.rtwh-aachen.de/~roberts/meshgeneration.html`

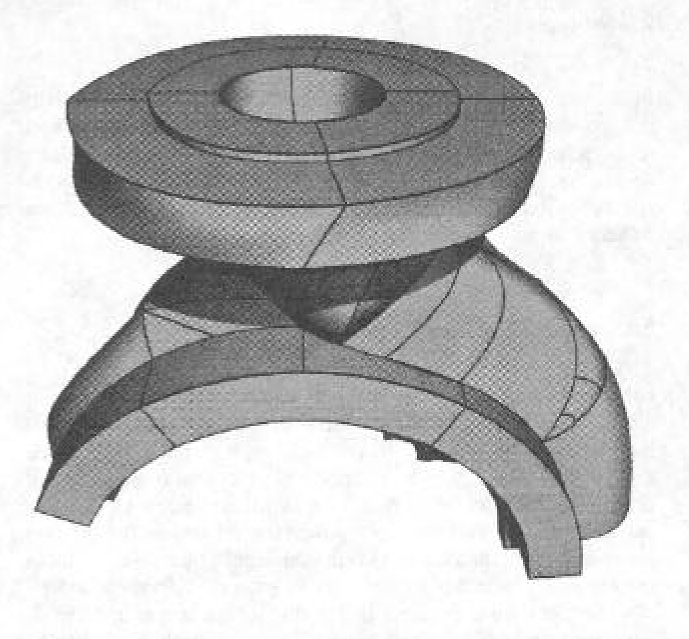
Figure 1: A CAD model of a pump, designed by an engineer in a German car company.

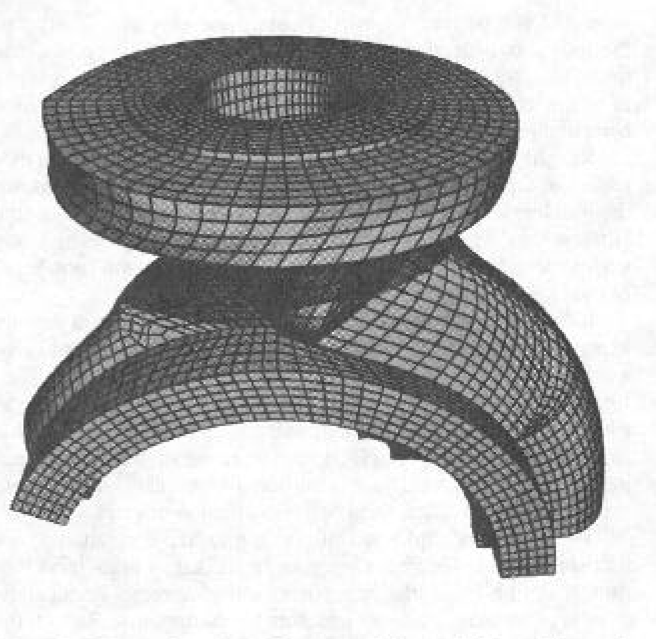
Figure 2: The conforming refinement of the model from Figure 1 generated by the algorithm from [13]. Obviously, the overall structure of the mesh is highly regular.

Obstacles. First we consider *the Formalization Obstacle.*

In principle, the goal that every quadrilateral in the refinement should be roughly square-shaped can be expressed in well-defined terms such as internal angles and so-called *aspect ratios (the* ratio of the lengths of opposite sides). However, the exact impact of these criteria on the solution quality is not at all understood: does it suffice to avoid extremely bad angles and aspect ratios, or is the average quality over all quadrilaterals more important; does the location of badly-shaped quadrilaterals make a difference; how important is it to avoid clusters of badly-shaped quadrilaterals? Various questions of this kind have not been answered satisfactorily so far. Hence, though certainly important, statistics on criteria like these are of a rather limited meaningfulness.

Even worse, it is not at all clear how to put the purely intuitive notion of *mesh regularity* in formal terms. One possibility discussed in the literature is the degree of mesh nodes. In that, a degree of four is regarded as optimal. However, analogous questions come up: does it suffice to avoid extremely high degrees; is the average degree more important; what about locations and clusters of bad mesh nodes? Even if these questions were answered satisfactorily, it is obvious that the node degrees do not really capture the intuitive sense of regularity.

Two further ideas to capture the regularity of a mesh are sometimes mentioned: symmetry and contour lines. Clearly, as a quality criterion, every kind of symmetry is of limited applicability. The imagination behind the other idea, contour lines, is that the ideal directions of mesh edges would be locally chosen parallel/orthogonal to the expected direction of the vector field that models the crucial physical phenomenon (e.g. local direction of power or heat flow). The goal is then to find a refined mesh such that the edges of the quadrilaterals roughly conform to these two directions. However, again it is not clear how to balance penalties for different kinds of deviations from such an ideal mesh.

The above-mentioned templates *are* an example of the *Maintenance/Volatility Obstacle.* In fact, the set of templates *did* change during the course of our project.

Finally, we consider the *Data Obstacle.* The $\mathcal{NP}$-completeness proofs [13] rely on certain classes of worst-case instances. However, these instances are highly pathological, so there is no hope that an experimental study based on them may be meaningful for real-world instances. On the other hand, we do not see any perspective to define a class of realistic random workpieces either (what is a realistic “random pump”?).

Framework. The following discussion is based on [12], [13], and [14]. Details are taken from [14].

The main idea is to abstract from the concrete geometric and numerical details as far as possible and to reduce the problem to a *combinatorial core problem.* In fact, our mesh-refinement problem can then be modeled as a capacitated *minimum-cost bidirected-flow problem* [7] in a certain variant of the dual graph of the mesh. Roughly speaking, the condition that the mesh be conforming can be translated into a certain kind of *flow conservation conditions*, which turns out to be a special case of the general bidirected flow conditions.

The intuition behind this translation is this: every quadrilateral in a *conforming* mesh (Figure 2) may be viewed as belonging to two *strings* of quadrilaterals. For a quadrilateral Q, let S_1 and S_2 be two opposite sides and S_3 and S_4 the other two opposite sides. Then one of the strings containing Q also contains the two quadrilaterals incident to S_1 and S_2, and the other string also contains the two quadrilaterals incident to S_3 and S_4.

Each such string in a conforming refinement like in Figure 2 corresponds to a path in the dual graph of the input mesh in Figure 1, and these paths can be summed up to a “flow” in the dual graph. The exact definition of this flow turns out to be a special case of the bidirected flow definition.’

[8]Under some circumstances an almost-all qudrilateral mesh, which contains a small number of triangles, is also acceptable or even necessary (see Figure 3(e)). In [14], we show that this case is also covered by our framework.

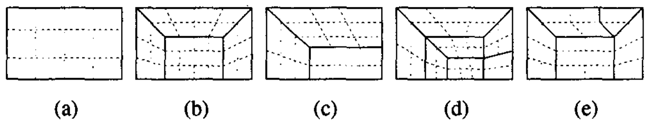

Figure 3: Typical examples of templates for the refinement of quadrilaterals into quadrilaterals. Part (e) shows an example of a template that deviates from the pure quadrilateral style and produces an isolated triangle.

In this flow model, each edge is assigned three values: an upper and lower capacity and a cost value. These three edge vectors allow a fine-grained, local control over various aspects. In [14], we describe how angles, aspect ratios, contour lines, local densities, and various further criteria can be controlled exclusively in terms of these vectors. Each criterion may be modeled as a "soft" or "hard" constraint, namely exclusively through the cost values or through cost values *and* capacities. If all criteria are modelled as "soft," an optimal solution is guaranteed in (low-degree) polynomial time.[9]

Since all relevant degrees of freedom are expressed as numerical values, it is technically easy to adjust all "screws" within this framework by experiments on benchmark instances. In principle, each criterion may be expressed by a large range of possible value settings in these three vectors, which express different possibilities to penalize deviations from the ideal realization of this criterion. Of course, these possibilities are not equally suitable. However, since everything is expressed in terms of three edge vectors, a suitable setting can be found by experiments. Moreover, the simultaneous incorporation of different aspects may be simply expressed by a superimposition of the corresponding triples of vectors (*e.g.* a component-wise weighted mean value). Hence, a good balance between different aspects can also be found experimentally.

Most "volatile" details (*Maintenance/Volatility Obstacle*) only affect the values of these three vectors. So an adaptation to a new problem variant might be manageable (even for non-experts in computational geometry).

Validation. We performed all of our experiments on real-world data from the German automobile industry.

In our experience, even the eye of a non-expert is quite good in estimating the quality of a mesh when presented graphically like in Figure 2 (provided it is a real-world mesh and not an artificial instance). In fact, the crucial characteristics of good meshes seem to be closely correlated with the human sense of aesthetics (recall Footnote 6). Due to this surprising insight, a domain expert can indeed train our algorithm on a given set of test data.

[9]If certain criteria (*e.g.* template restrictions) are incorporated as "hard," the problem becomes $\mathcal{NP}$-hard.

2.2 CAD Data Repair

Introduction. The input is a CAD model as described in Section 2.1 (see Figure 1). More specifically, such a model is only given as an unstructured set of polygons. The algorithmic problem here is to reconstruct the information which polygons are to be regarded as neighbored (*incident*) and in which order the neighbored polygons appear around each polygon.

This information is usually called the *topology* of the model, and the reconstruction of the topology is another crucial preprocessing step in the CAD process. Although the problem appears in the same application domain as the mesh-refinement problem from Section 2.1, it is obviously very different by nature.

Figure 1 suggests that two neighbored mesh elements always meet geometrically. However, in general, two mesh elements that are intended to be neighbored are only located (more or less) close to each other in the three-dimensional space. Figures 4 and 5 demonstrate this fact. It is this subtlety which makes the problem nontrivial.

A mesh of a hollow body may also contain *intended* holes, namely wherever the modeled hollow body has an opening. So we may reformulate the problem as follows: *the problem is to distinguish unintended gaps between neighbored elements from intended holes inherent in the workpiece.*

For example, unintended gaps may be due to errors in the interactive design of a mesh or caused by conversions between incompatible data formats. The on-going trend towards simplistic standard exchange formats will make the task of reconstructing the topologies of faulty meshes even more urgent in the future.

To our knowledge, all previous work is mainly based on the following straight approach: a distance function on pairs of polygons is defined (*e.g.* the minimal Euclidean distance), and a certain threshold value is chosen. Two polygons are regarded as neighbored if and only if their distance is smaller than this threshold value. However, the typical scenario in the literature is quite different, namely that every mesh approximates the surface of a solid three-dimensional body. In this scenario, intended holes do not appear, so the main problem of *our* scenario simply vanishes.

Obstacles. The information to be reconstructed is the distinction between intended holes and unintended gaps. In other words, this is an example of an algorithmic problem with inherent subjectivity as described in the *Formalization Obstacle* in the Introduction.

This evidence is supported by an empirical study [21]. The main outcome of this study is the conclusion that the above-mentioned approach (a distance function and a distinguishing threshold value) is probably generally inadequate for real-world data like the ones available to us.

More specifically, we evaluated a broad range of distance functions, which might sufficiently cover all natural definitions. It turned out that none of these distance functions admits a threshold value that distinguishes correct from incorrect neighborhoods with an acceptable number of errors. In fact, the data is much "dirtier" than suggested by pictures like Figure 1. For instance, the interior parts of the models are much dirtier than the outer parts (as in Figure 5).

We also looked for "patterns" in the statistics, which could give a hint to better formalizations. However, we failed. In our opinion, this gives strong evidence that the inherent subjectivity cannot be disregarded and does not allow a sufficiently simple mathematical model.

For the same reasons as in Section 2.1, the *Data Obstacle* is a serious handicap. Even worse, here an artificial instance generator would also have to generate "artificial intended holes" and "artificial unintended gaps." However, any artificial definition of holes and gaps would inevitably reflect our (limited) understanding rather than the holes and gaps in real instances.

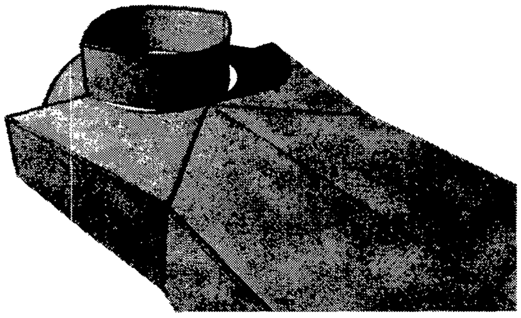

Figure 4: A significant example of gaps that are obviously unintended.

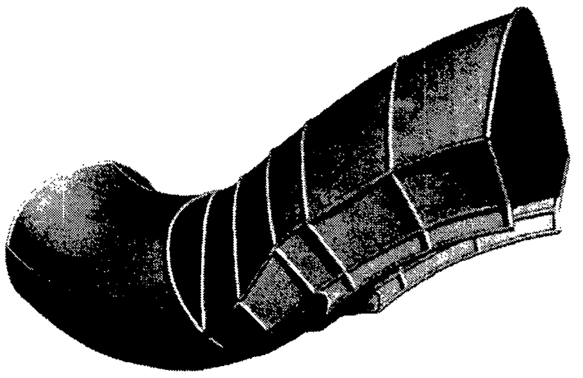

Figure 5: An excerpt from inside the pump shown in Figure 1. Only the additional background knowledge about the purpose of the workpiece proves that the black semicircular hole has to be regarded as an unintended gap.

Framework. In the case of surfaces of solid bodies, we may rely on a strong assumption: the mesh is intended to form a single, closed, orientable manifold. However, for hollow bodies, all we know in formal terms is that the mesh should form a set of openly disjoint, two-dimensional manifolds in the three-dimensional space.[10] A hollow body may be internally structured by dividing walls, rims, and the like, so the composition from manifolds may be quite complex.

Nonetheless, in [20], we have demonstrated that the intuitive insights into the nature of real-world instances are strong enough to develop certain *logical inference rules*, which may well detect and resolve impossible local configurations. A simple, illustrative example is transitivity of neighborhoods: if a polygon P is neighbored to polygons P_1 and P_2 and if the boundary segment of P neighbored to P_1 significantly overlaps with the boundary segment of P neighbored to P_2, then P_1 and P_2 also have to be regarded as neighbored. (Such a configuration occurs whenever different manifolds meet.) Formulated the other way round: if we know that P_1 and P_2 are not neighbored, then P cannot be neighbored to both P_1 and P_2 simultaneously via mutually overlapping segments.

This is but one illustrative example of possible logical inference rules. See [20] for further details.

In our approach, these logical inference rules are applied to a systematic overestimation of the real neighborhoods. Hence, an impossible local configuration is resolved by removing one of the involved neighborhood relations in the current, overestimating candidate set. This approach gives two major degrees of freedom: how to compute the initial overestimation and which candidate to remove in case an impossible configuration is detected. Moreover, some of the inference rules are parameterized by numerical values, which also serve as "little screws."

Roughly speaking, the initial overestimation is computed as follows. We select a finite set of straight lines and project each polygon orthogonally onto each of these straight lines. For every pair of polygons we count the number of straight lines on which their projections overlap. If this number is too small, this pair of polygons is disregarded henceforth. An appropriate choice of straight lines and of a minimal number of overlaps can quite well be determined experimentally, based on the quality of the output computed for benchmark instances.

To our surprise, the above-mentioned distance functions have turned out to be quite suitable for the decision which candidate to remove in case of an impossible configuration: just remove the candidate of highest distance. Hence, in our concrete application this particular degree of freedom need not be adjusted experimentally. Other applications may fall back to this opportunity if necessary.

Validation. Our aim at an insightful visualization has significantly influenced the design of the algorithm. More specifically, the following observation guided the design: if the output of the algorithm is still a systematic overestimation of the real result, then a coloring of all mesh edges according to the number of incident polygons unambiguously reveals all errors in the output to a human observer.[11] See Figure 6 for a simple example.

In particular, we did not tune the individual degrees of freedom so as to minimize the number of errors; we rather tuned them to minimize the number of errors *subject to a side constraint*, which reflects the above discussion: that the result is guaranteed to be an overestimation.

Hence, this project nicely demonstrates a point we emphasized in the Introduction (see "Intent of the paper"): a meaningful visualization does not come for free and was hence treated as a first-class algorithmic design goal.

[10]Even these weak conditions are not really guaranteed. However, for ease of exposition we will disregard this point in the following discussion.

[11]For example, if the output is *not* an overestimation, the following kind of error, which may often occur in the interior of a complex-structured hollow body, cannot be detected by such a coloring: polygons P_1 and P_2 are intended to be neighbored and likewise polygons P_3 and P_4, however, the algorithm may instead deliver a neighborhood relation between P_1 and P_3 and between P_2 and P_4. Hence, the result is wrong, however, the visualization displays the same colors as if the correct neighborhoods were found.

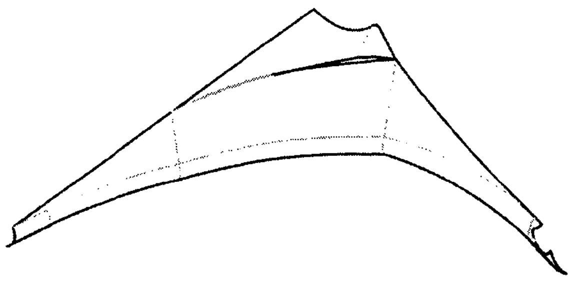

Figure 6: An edge is colored black if it is incident to exactly one polygon. This simple visualization technique reveals that the obviously unintended gap was erroneously regarded as an intended hole.

2.3 Inferring a Railroad Network from Time Tables

Introduction. We have the problem of inferring the structure of a railroad network when only time tables of trains operating on that network are given.

We are given train time tables of long distance, regional, and local trains of most European countries consisting of more than 140 000 trains. Together, they stop at about 25 000 train stations all over Europe which are given with coordinates describing their geographical locations. Each time table contains a list of consecutive stops for one particular train, indicating the times of arrival and departure for each stop, as illustrated by the following excerpt from a time table for a train through the Black Forest:

Offenburg		07:02
...		
Villingen	08:09	08:11
Donaueschingen	08:20	08:21
Immendingen	08:33	08:33
Engen	08:45	08:45
Singen	08:55	08:56
Radolfzell	09:03	09:05
Allensbach	09:10	09:11
Konstanz	09:19	

Figure 7 suggests that, on its way from Donaueschingen to Immendingen, the train passes through another train station without stopping there, while from Immendingen to Engen, there are no additional stations on the way that are merely being passed through. So the line between Immendingen and Engen in Figure 7 represents a segment of the physical railroad network, while the line between Donaueschingen and Immendingen does not. The problem to be solved is now to decide, for each (unordered) pair of consecutive stops appearing in some time table, whether it represents a segment of the physical railroad network or not. To our knowledge, this problem has not been studied before.

When looking at a visualization of time table data as depicted in Figure 7, the identification of the physical railroad network seems intuitively clear from the picture, because we can draw on our experience what physical railroad networks normally look like. But it is difficult to translate our intuition into precisely defined properties that yield an algorithm. Besides, we need to keep in mind that our intuition might fail: it is for example not clear that we would identify the line between Tuttlingen and Engen as belonging to the physical railroad network, because alternatively, the trains

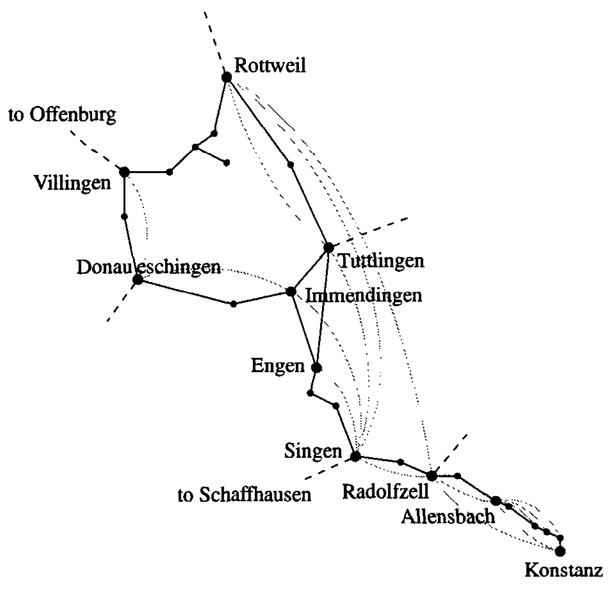

Figure 7: In order to visualize train time table data we place each train station according to its geographical location. Stations a and b are connected by a line if and only if a and b are consecutive stops of at least one train. In this example, pairs of consecutive stops representing the physical railroad network are drawn in black, and the other pairs of consecutive stops are drawn in grey.

with Engen and Tuttlingen as consecutive stops could pass through Immendingen.

The relevance of the *Data Obstacle* is empirically supported by the experiences gained from the treatment of another algorithmic problem on the same data (see Footnote 3).

Framework. Consider the undirected *time table graph* $G = (V, E)$ induced by a set of time tables: Each train station appearing in some time table is a vertex of the graph, and $\{a, b\}$ is an edge of the graph if and only if there is at least one train that has a and b as (an unordered pair of) consecutive stops. By definition, a time table graph has no multiple edges.

Since the train time tables are the only available information, we cannot but assume that every segment of the physical railroad network is also contained as an edge in the time table graph and hence part of the input. Given a set of time tables and the time table graph $G = (V, E)$ it induces, the problem is then to find the *physical railroad subgraph* $G' = (V, E')$ of G so that each edge in E' represents a segment of the physical railroad network, and so that each edge in $E \setminus E'$ does not.

As visualizations such as Figure 7 suggest, considering local properties such as Euclidean lengths of edges, angles between edges, or vertex degrees lead to the classification of edges as belonging or not belonging to the physical railroad subgraph. For a simple example, if a vertex has degree two and the incident edges are opposite of each other, then it is likely that these edges belong to the physical

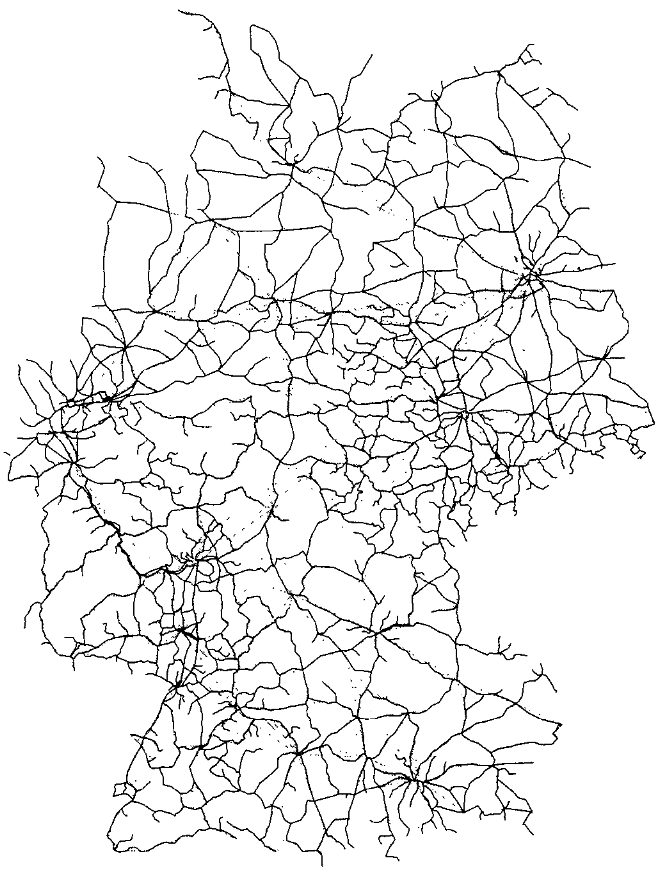

Figure 8: The time table graph induced by trains in Germany. The black edges indicate a heuristically determined physical railroad subgraph. Close inspection suggests that some grey edges actually should belong to the physical railroad subgraph but were not recognised by the heuristic.

railroad subgraph. How wide an angle between two such edges has to be for the edges to be considered opposite is one of the "screws" of a classification algorithm. Figure 8 shows a physical railroad subgraph determined heuristically using such local properties.

Our main approach, however, is based on the structure of the time table graph rather than on local properties. Observe that along a line of railroad tracks where the railroad network does not branch (in Figure 7 for example between Konstanz and Radolfzell), it is easy to identify the physical railroad subgraph edges: Since the sequence of train stations visited by trains is known from the time tables, the train stations along such a line of railroad tracks can be linearly ordered, and the edges connecting two vertices that are consecutive in this linear ordering are exactly the ones belonging to the physical railroad subgraph. Edges connecting two train stations where the physical railroad network branches (such as {Radolfzell, Rottweil} in Figure 7) remain unclassified with this approach.

Even though it is not clear how the branching points may be reliably identified, we can guess a set $V' \subseteq V$ of branching points and partition the edge set of the time table graph according to V'. For each edge set of the partition we can determine whether or not it does contain all edges along one line of railroad tracks, and if it does, we can easily classify each of its edges.

There are many degrees of freedom within this framework, for example in the way we guess a set of branching points for the structural approach, or in the way we combine the structural approach and the approaches based on local properties to classify as many edges of a given time table graph as possible.

Validation. One necessary condition that an edge classification algorithm is performing well is a high number of classified edges versus the number of edges of the graph, but we obviously have no way of checking the correctness of the classifications. We can only visualize the result of the algorithm (or, because of the size of the time table graph, parts of it) as in Figure 7 and inspect the picture to see whether it corresponds to our intuition what the physical railroad network should look like. For small geographical areas we can even compare the result to a conventional map. So a human being can check for small parts of a result whether it is, or at least looks, correct. As a consequence, the classification algorithm may be modified and the result inspected again and so on until a seemingly good classification algorithm has been obtained.

2.4 Social Network Visualization

Introduction. Social network analysis is a research methodology from the social sciences, in which social structures are modeled by means of a graph [19]. Relevant *actors, i.e.* persons, organizations or other social entities, are represented as nodes of the graph. Edges are formed by relations of affective, political, economic, interactional, organizational, or similiar type. Depending on the type of relation, the graph is directed or undirected. For example, the relation "influences" would naturally induce directed edges, whereas a symmetric relation such as "collaborates with" would induce undirected edges.

Social network theory is the attempt to account for the constraining influence of other actors in decision making. An observed network is therefore analyzed in order to explain the behavior of a set of actors, or particular outcomes of a decision making process in which they were involved. An example of an important research question is, which actors are "central" to the structure (see Figure 9), because these actors are assumed to be the most influential.

Both the exploration of social structures and the communication of results greatly benefit from appropriate visual presentation of the data [5]. For ease of exposition, we here restrict ourselves to the form most common for graphical presentation of networks, that is to represent each node by a point, and each edge by a connecting straight line segment. The layout problem then reduces to the positioning of nodes. Experiments show that node positioning has a significant impact on the perception of structural properties [2].

Obstacles. First we consider the *Formalization Obstacle.*

The variety of possible aspects that can make a particular network interesting to an analyst is immense, and often closely related to the specific context of the network. When exploring network data, the researcher cannot be assumed to precisely know in advance what he or she is looking for. Therefore, a visualization suggesting unfounded properties may easily misguide the researcher's intuition. The ultimate goal would be a drawing that reveals exactly the essential structural properties inherent in the network, without distortion. Hence, this problem is another example of aspects that are difficult to capture formally, and highly interrelated.

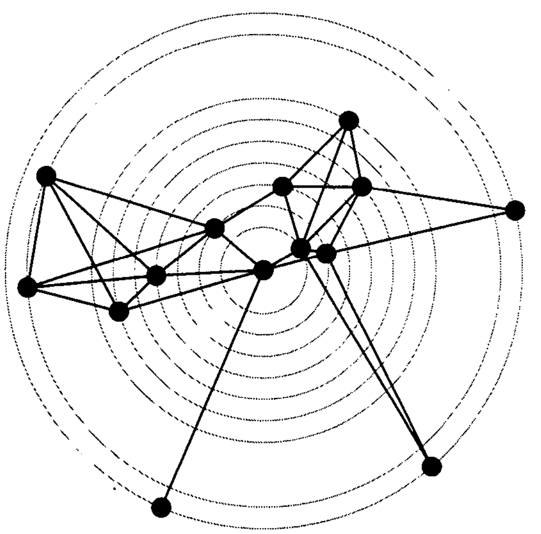

Figure 9: Visualization of a network of political actors (data taken from [6]). Each node represents a person, and two persons are connected by an edge if they have strong political ties. The radius onto which a node is positioned is determined by the sum of distances to all other nodes (so-called *closeness centrality*). The radians were determined through interaction potentials that penalize long edges, vertex-vertex and vertex-edge overlaps, and edge crossings.

Several structural variables are employed in an analysis. Of course, a strong focus on one variable may cause a distortion in the display of others. When a user asks for a more accurate representation of a particular aspect, he/she is often dissatisfied because the simultaneous understanding of other properties is not necessarily supported anymore. We have to be prepared that the inevitable compromise is subject to change (*Maintenance/Volatility Obstacle*).

Since it is important to have a sufficiently large set of test networks, we are also faced with the *Data Obstacle*. In order to evaluate the quality of a visualization, it is important to have a good understanding of the conclusions that an observer *should* draw. The data used in Figure 9 is one of the rare examples that have been analyzed over and over again. Quite conversely, experiments on artificial data would reveal the properties that we deliberately put in the instance generator and could not be cross-checked.

Framework. To protoype our layout models, we make use of a very general framework. We briefly outline its specialization to straight-line representations of undirected graphs $G = (V, E)$. See [4] for a more elaborate description and a different specialization. Here, each node $v \in V$ is assigned a position x_v from a feasible set $\mathcal{X}_v$. In the case of Figure 9, $\mathcal{X}_v$ is formed by the respective circle. Any layout of the network is fully described by a vector from the Cartesian product of all $\mathcal{X}_v$, $v \in V$.

Criteria for good layout are formulated locally by *interaction potentials* defined on subsets of V. These potentials depend on the positions of nodes in their associated subset, but not on the position of any other node. They are additively combined into an objective function evaluating the total layout quality. For example, the criterion that an edge $\{u, v\} \in E$ should have a certain length can be expressed by measuring the deviation of the Euclidean distance between x_u and x_v from this target length. Desired edge lengths are hence local criteria defined on two-element subsets of V. Similarly, vertex-edge distances are formulated in terms of three-element subsets of v, and edge crossing can be counted on four-element subsets. Using parameters and weights, these criteria can be scaled.

This framework generalizes a number of approaches known from the literature. The selection and parametrization of interaction potentials are the essential degrees of freedom ("screws") in our model. Parameters of the potentials used in Figure 9 have been adjusted experimentally so as to yield a readable layout.

Validation. In the first phase of our (on-going) project, we have chosen the important operational concept of *centrality* and a concentric style of presentation as shown in Figure 9. This style proved to highlight the relative centrality of each actor better than other styles we tried out. A number of classical data sets from the literature and new ones from our associates (researchers in political science) are used to experiment with different variants of centrality-oriented layout. Feedback is incorporated by matching the information conveyed by our layouts with known substance of these networks and results of quantitative analysis. Together with cognitive psychologists, more systematic experiments are planned to validate our preliminary results.

3 Summary

In the four projects discussed above, the obstacles to a precise mathematical problem formulation were quite versatile.. For example, we have seen that subjective aspects cannot always be disregarded. However, even if the problem is purely objective, it may be too complex to allow more than a *very* rough approximation. Anyway, an appropriate reduction to a set of formal rules does not seem to be in our reach.

Nonetheless, it has turned out that a common perspective is quite promising: instead of a detailed problem definition, a "problem framework" was defined, which spans a large range of possible tailored definitions but also allows a very "fine-grained" customization: due to the various numerical "screws," it should be possible to come quite close to the real problem because these screws allow the incorporation of aspects that resist the reduction to explicit, formalizable criteria.

Since the real problem definition is not known in formal terms, we need something beyond that. This is the reason why each of these projects "discovered" the power of human interaction and its support by appropriate visual aids. This is not surprising, since visual aids are generally a favorable means of presentation. It has turned out that intuitive, "unconscious" domain knowledge can indeed be utilized on such a visual basis, namely to iteratively approach a good adjustment of the "screws."

It seems that such a generalized framework is much simpler and thus easier to understand by non-experts than a tailored, sophisticated problem definition, because many difficult details do not explicitly appear in the problem definition but implicitly, in the values chosen for the degrees of freedom, and thus do not require a deep understanding from the maintainer.

In particular, the repetition of the iterative customization process should also be feasible for non-experts in computational geometry, so chances are high that maintenance is successful even in case the logic of the algorithmic problem is affected by "volatile" details of the application domain (see the *Maintenance/Volatility Obstacle* in the Introduction).

In such an iterative, empirical convergence process towards a problem definition, the need for real-world data (*Data Obstacle*) is even more virulent than in other approaches: it seems that a domain expert needs certain characteristics of the real-world data to fit the visual presentation of the output into his or her intuition of the problem. Artificial classes of instances do not necessarily capture these characteristics, because an artificial instance generator may only reflect the characteristics understood by its creator.

The topics discussed in this paper are certainly not restricted to computational geometry. However, the individual projects have demonstrated that these topics are relevant in various settings which are more or less close to computational geometry. Even more, the strong connection to visualization makes them geometric topics in the first place.

References

[1] M. Bern and D. Eppstein: *Mesh generation and optimal triangulation.* Computing in Euclidean Geometry, 2nd Edition (D.-Z. Du and F. Hwang, eds.), World Scientific, Singapore, 1995, pp. 47–123.

[2] J. Blythe, C. McGrath, and D. Krackhardt: *The effect of graph layout on inference from social network data.* In F.J. Brandenburg (ed.): Proc. 3rd Int. Symp. Graph Drawing (GD '95), LNCS 1027, 40–51.

[3] T. D. Blacker and M. B. Stephenson: *Paving: A new approach to automated quadrilateral mesh generation.* Int. J. Numerical Methods in Engineering **32** (1991), 811–847.

[4] U. Brandes and D. Wagner: *Using graph layout to visualize train interconnection data.* Proc. 6th Int. Symp. Graph Drawing (GD '98), LNCS 1547, 44–56.

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

[6] P. Doreian and L.H. Albert: *Partitioning political actors: Some quantitative tools for analyzing qualitative networks.* J. Quantitative Anthropology **1** (1989), 279–291.

[7] B. Gerards: *Matching.* In M. Ball et al. (eds.): *Network Models.* Handbooks Operations Research and Management Science **7**, 135–224. Elsevier, 1995.

[8] K. Ho-Le: *Finite element mesh generation methods: a review and classification.* Computer-Aided Design **20** (1988), 27–38.

[9] T. Lengauer: *Bioinformatics—a different algorithmic experience.* Slides of an invited lecture, On-line Proc. 2nd Workshop Algorithms Engineering (WAE '98).[12]

[10] S. Mitchell: *Choosing corners of rectangles for mapped meshing.* Proc. 13th ACM Symp. Computational Geometry (SCG '97), 87–93.

[11] B. Moret: *Experimental algorithmics: old problems and new directions.* Slides of an invited lecture, On–Line Proc. 2nd Workshop Algorithms Engineering (WAE '98).[12]

[12] R. H. Möhring and M. Müller-Hannemann: *Complexity and modeling aspects of mesh refinement into quadrilaterals.* Proceedings of the 8th Int. Symp. Algorithms and Computation, ISAAC'97, LNCS **1350**, Springer Verlag, pp. 263-273, 1997.

[13] R. H. Möhring, M. Müller-Hannemann, and K. Weihe: *Mesh refinement via bidirected flows: modeling, complexity, and computational results.* J. ACM **44** (1997), 395–426.

[14] M. Müller-Hannemann and K. Weihe: *On the discrete core of quadrilateral mesh refinement.* To appear in the Int. J. Numerical Methods in Engineering.

[15] Thomas M. Pigorski: *Practical software maintenance.* Wiley, 1997.

[16] T.K.H. Tam and C.G. Armstrong: *Finite element mesh control by integer programming.* Int. J. Numerical Methods in Engineering **36** (1993), 2581–2605.

[17] G. Toussaint: *Quadrangulations of planar sets.* Proc. 4th Int. Workshop Algorithms and Data Structures (WADS '95), LNCS **955**, 218–227.

[18] K. Weihe: *Covering trains by stations or the power of data reduction.* On-Line Proc. 1st Workshop Algorithms and Experiments (ALEX '98).[13]

[19] S. Wasserman and K. Faust: *Social network analysis: methods and applications.* Cambridge University Press, 1994.

[20] K. Weihe and T. Willhalm: *Reconstructing the topology of a CAD model – a discrete approach.* Proc. 5th European Symp. Algorithms (ESA '97), LNCS **1284**, 500–513 (journal version to appear in Algorithmica).

[21] K. Weihe and T. Willhalm: *Why CAD data repair requires discrete techniques.* On–Line Proc. 2nd Workshop Algorithms Engineering (WAE '98).[12]

[12] http://www.mpi-sb.mpg.de/~wae98/PROCEEDINGS/

[13] http://rtm.science.unitn.it/alex98/proceedings.html