Algorithms for the Placement of Diagrams on Maps*

Marc van Kreveld

Inst. of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

marc@cs.uu.nl

Étienne Schramm‡ Alexander Wolff‡

Fakultät für Informatik, Universität Karlsruhe
P.O. Box 6980, 76128 Karlsruhe, Germany

http://i11www.ira.uka.de/people

ABSTRACT

This paper discusses a variety of ways to place diagrams like pie charts on maps, in particular, administrative subdivisions. The different ways come from different models of the placement problem: a diagram of one region should cover other regions, roads or boundaries as little as possible. In total we present six models for diagram placement. We outline three different algorithmic approaches and discuss the efficiency of each approach for the different models, and also for different types of diagrams (rectangular, circular, same or different sizes). We have implemented an algorithm for each model and show the resulting diagram placements on a number of maps. Our evaluation gives a first indication which model is best for aesthetically good diagram placement.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

General Terms

Algorithms, Experimentation

*This work was started during the fourth seminar on *Computational Cartography and Spatial Modelling* at the International Conference and Research Center for Computer Science, Schloss Dagstuhl, 28.09.–03.10.2003.

‡Supported by grant WO 758/4-1 of the German Science Foundation (DFG).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GIS'04, November 12–13, 2004, Washington, DC, USA.
Copyright 2004 ACM 1-58113-979-9/04/0011 ...$5.00.

Keywords

Placement algorithms, cartographic visualization

1. INTRODUCTION

Maps are mostly used for general reference, such as finding routes and locations. However, maps are also used to visualize additional information on various regions. For example, when the regions are countries, one may want to visualize data like unemployment of each country. When a shade of some color is used for each country, this is called choropleth mapping. Other options are to annotate the regions of the map with the data. This is particularly useful when the theme to be mapped is more complex and cannot be represented by a single value per region. If the work labor divided into agricultural, governmental, and industry is mapped per country, then a pie chart or histogram can be placed in each country. This is a common mapping technique for human geographic and economic geographic data. Examples can be found in cartography textbooks (Dent 1999, Robinson et al. 1995) and in visual almanacs (Glassman 1996).

This paper discusses the problem of placing diagrams like pie charts into regions. The labels are of fixed size, and we want to avoid overlap between different labels. It is a special case of the so-called *map labeling problem*. There is an extensive bibliography (Wolff and Strijk 1996) that lists publications concerned with algorithms for this problem and its special cases.

Our problem is related to text placement for areal features, but it differs from it in major points: the dimensions of the label, as well as its orientation and its shape, are given *a priori*, whereas the text labels may be scaled and rotated in order to be visually more pleasant. Another difference is that the labels are supposed to be totally opaque, and to completely hide what lies under them, whereas in the case of text placement there may be some space between letters. Most modern methods for text placement take these possibilities into account. Pinto and Freeman (1996) use an iterative feedback approach that is guided by some heuristics (most of them are irrelevant in our case), that often leads to a visually pleasant label that is curved to suit the shape of the region. Edmondson et al. (1997) also use heuristics,

but the method is simpler: a large number of quasi-random candidates are generated and the best one according to the heuristic is chosen. The authors also consider the case that the text label does not fit in its region. In that case, they define a point representing the region and use a point labeling algorithm.

A recent paper on icon placement on maps (Harrie et al. 2004) is related too. Here the objective is to cover as few points of interest as possible and to preserve the map's overall legibility. Although the context is different (icons generally refer to points of interest and the density of icons is usually smaller than in our case), the main aims are, as in our case, to minimize the loss of visual information and to avoid overlap between labels.

There are few papers that consider explicitly the placement of labels of a given size in a region. Van Roessel (1989) computes a set of maximal rectangles contained in the region and chooses one of them, so the label has to fit completely in its region. Arikawa and Kambayashi (1991) remove this restriction, but the position of the rectangle entirely depends on the shape of its own region, and does not take into account neither other regions, nor the shape of the label. In fact, they present a straightforward extension of traditional proportional symbol placement, in that they define a point (that differs from the centroid) that depends only on the shape of the region to be labeled, and they put the center of the label at this point.

Our approach differs from traditional approaches to proportional symbol placement in that we try to have as little overlap as possible between two labels. In proportional symbol placement, it is usually not important (and often not suitable) that the symbols do not overlap, as long as a part of each one can be seen. On the other side, if there is some additional information on the labels, such as textual information, any overlap is a problem.

High-quality diagram placement immediately raises the issue of how the problem should be formalized, that is, which criteria should be taken into consideration and how. As said before, the most important criterion for us is that no two diagrams (belonging to different regions) overlap. If a diagram fits inside its region, then it should be placed in or near the center. If a diagram does not fit inside its region, then it should be placed so that it mostly overlaps with its own region (to obtain clear association between region and diagram), but avoids overlap with the region boundaries as much as possible (to maintain recognizability of the regions).

Even if we only consider the first criterion, the problem of placing equal-sized diagrams without overlap in a bounded map area (possibly outside their region), we are faced with a computationally hard problem, namely that of *packing* (Tóth 2004). However, in practice the problem is not so difficult because the total area taken up by the diagrams is only a small part, say, 10–40%, of the total map area. Hence, we will not attempt to address and solve the problem theoretically.

The straightforward approach of centering every diagram at the center of gravity of its region is a practical solution that will often give good results. However, to achieve high-quality results without manual post-processing, it will not be good enough. For example, if two small countries are neighboring, then their diagrams can overlap (e.g., states Vermont and New Hampshire; Figure 1), or a diagram may be unnecessarily much outside its own region (e.g., the country Israel). The center-of-gravity approach is for example used by GIS tools for proportional symbol placement. In this context, it leads to good results since the aim goal is not to avoid overlaps.

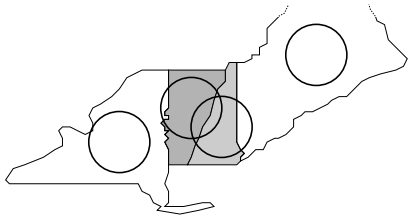

Figure 1: Placement at the center of gravity is not always a good idea.

This paper discusses various models for placing a pie chart or rectangular diagram with a region without explicit regard to the placement of other charts and diagrams. We will optimize area overlap with the region to be annotated, minimize length overlap with boundaries, or minimize area overlap with small neighboring regions. Especially the last criterion (implicitly) helps to avoid intersections between diagrams.

The remainder of this paper is organized as follows. Section 2 discusses the models for diagram placement that we consider. Section 3 describes the algorithms and their efficiency for each of the models. We implemented one algorithm for each model. In Section 4 we show examples of maps with diagrams that our algorithms placed. This makes it possible to compare the effects of the different models. Efficiency of the computation and aesthetics of the output help to decide which model is best in which situation. In Section 6 we conclude and list possible directions for further research.

2. MODELS OF DIAGRAM PLACEMENT

We consider the following six models of the diagram placement problem for a subdivision S into m regions $R_1, \ldots, R_m$ to be annotated. Each region is assumed to be a simple polygon. The diagram C_i to be placed for region R_i can be circular or rectangular, and has a pre-specified size. Its center is denoted by c_i.

1. ***Centroid***: Place the center c_i of the diagram at the center of gravity of R_i.

2. ***MaxSelfOverlap***: Place the diagram C_i such that the area of overlap with R_i is maximized.

3. ***MinAreaOverlap***: Place the diagram with $c_i \in R_i$, and such that the area in other annotated regions is minimized. This is useful to allow partial placement in seas, oceans, and other irrelevant regions. In a formula:
$$\min_{c_i \in R_i} \text{area}(C_i \cap \bigcup_{j \neq i} R_j).$$

4. ***MinBorderOverlap***: Place the diagram with $c_i \in R_i$, and such that it covers the smallest possible total edge length of subdivision S.

5. ***MinMaxOverlapPerc***: Place the diagram with $c_i \in R_i$, and such that the percentage that it covers of the

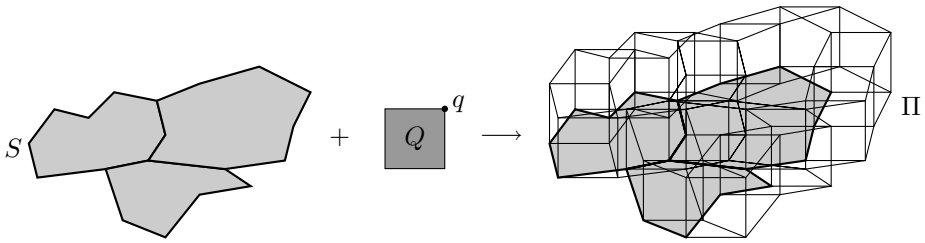

Figure 2: Left: a simple subdivision S with three regions. Right: the arrangement Π representing the combinatorially distinct placements of the square.

area of any other region is minimized. In a formula:

$$\min_{c_i \in R_i} \max_{1 \le j \le m,\ j \ne i} \text{area}(C_i \cap R_j)/\text{area}(R_j)$$

6. ***MaxMinUncovered***: Place the diagram with $c_i \in R_i$, and such that the free area of every region of S except for R_i which intersects the diagram is maximized. In a formula:

$$\max_{c_i \in R_i} \min_{\substack{1 \le j \le m, j \ne i \\ R_j \cap C_i \ne \emptyset}} \text{area}(R_j) - \text{area}(C_i \cap R_j),$$

where the minimum is meant to return ∞ in case no region R_j intersects C_i.

Model *MinMaxOverlapPerc* states for example that it is much worse to cover 1 cm^2 of a small region than of a larger region, since we measure by percentage of area covered. Model *MaxMinUncovered* is to only consider regions that intersect the diagram, and of these, make sure that after placement the region with least area still has as much area as possible. If possible, any intersection of the diagram with a small region will be avoided.

All models but *Centroid* and *MaxSelfOverlap* specify that the center of the diagram should be inside R_i, otherwise all diagrams could possibly be placed in the sea. We could have resolved this problem in a different way, but we chose this condition because of its simplicity. We could also have included the criterion of avoiding covering the points where three regions meet in the subdivision. In some of the models this condition can easily be included.

In model *MinBorderOverlap* we attempt to place the diagram with minimum length overlap with the region boundaries. We can, however, easily include other linear features that should be avoided as much as possible. This is important in various situations. For example, assume we want to design a map with the German Bundesländer and their highways. We want to show the whole highway network plus various highway statistics for each Bundesland in, say, a rectangular diagram. We want to place the rectangles inside the appropriate Bundesland (for the larger part), while intersecting as little of the highway network as possible. The subdivision into Bundesländer is used to restrict the positions for the rectangles, while the highway network (possibly together with the Bundesland boundaries) is used to find minimum overlap placements.

If a region is sufficiently large and well-shaped so that it can contain its own diagram, then generally there are many different diagram placements that are equivalent with respect to the models. That is, the minimization problems can yield a value of 0 for several different diagram placements. In such a case the diagram can best be placed to maximize its distance to the boundary of its region. This is both aesthetic and implicitly avoids overlap with other diagrams that may cover part of this region near the boundaries.

3. ALGORITHMS FOR DIAGRAM PLACEMENT

In this section we describe algorithms for the placement of diagrams of prescribed size. We concentrate on square charts. This is equivalent to placing rectangular charts of fixed aspect ratio. Each square must be placed parallel to the axes at a position to be computed. Efficient algorithms for the models given are described in the next subsections. For model *Centroid*, we simply note that the center of gravity of a polygon with n vertices can be computed in linear time. Hence, we only discuss the other five models. The placement of circular charts is discussed at the end of this section.

In the efficiency analysis, we use n to denote the total number of vertices of the subdivision S, and m to denote the number of different regions.

3.1 Combinatorially distinct diagram placements

We will first analyze how many distinct placements exist for a square Q and the subdivision S. With "distinct" we mean that the subset of edges of S that Q intersects is different. If we imagine Q translating over S, then a different subset of edges is obtained exactly when one of the following occurs:

- an edge of Q passes over a vertex of S;
- a vertex of Q passes over an edge of S.

If Q translates and neither of these events occurs, then Q keeps intersecting the same subset of edges of S. We will define a configuration plane Π that describes the distinct placements of Q. Let us choose the upper right corner of Q as the reference point q of Q. The configuration plane Π should have cells such that q is inside a cell when Q intersects only one subset of the edges of S. Figure 2 shows a subdivision S, a square Q, and the resulting configuration plane Π with its partitioning into cells. The reason for the partitioning is the following. Within one cell of the arrangement, the area of overlap of the square and some region is described by a quadratic function $f(x, y)$. This quadratic function is needed for the optimization, and it is different in

every cell. The next lemma gives a quadratic upper bound on the number of distinct placements of Q.

LEMMA 1. *Let E be the set of n edges of a planar subdivision, and let Q be a square of fixed size and orientation. There are $O(n^2)$ distinct subsets of edges of E intersected by Q for the placements of Q.*

We show that the arrangement in the configuration plane can be constructed efficiently. The arrangement in Π is induced by a set of $O(n)$ line segments and can be constructed by a plane sweep in $O((k+n)\log n)$ time (Bentley and Ottmann 1979, de Berg et al. 2000), where k is the number of intersections. The resulting subdivision has complexity $O(n+k)$, which in practice is less than quadratic. More complex algorithms can construct the arrangement induced by a set of n line segments in optimal $O(k+n\log n)$ time (Chazelle and Edelsbrunner 1992, Mulmuley 1993).

THEOREM 1. *Given a connected subdivision S with n edges and a square Q, the arrangement representing all distinct placements of Q with respect to S can be constructed in $O(n\log n+k)$ time, where $k=O(n^2)$ is the number of distinct placements.*

We can extend this arrangement so that for each cell, if the reference point is in that cell, then the center of the square lies inside only one of the regions. In the models *MinAreaOverlap*, *MinBorderOverlap*, *MinMaxOverlapPerc* and *MaxMinUncovered*, this implies that for any cell of the partitioning it is uniquely defined which region from $R_1,\ldots,R_m$ can place its square with the reference point in that cell. The extended arrangement still has $O(n^2)$ complexity.

The same approach can be used for circular diagrams. The configuration plane Π is partitioned by a set of $O(n)$ circles and line segments into quadratically many cells. All circles whose reference points lie in the same cell intersect the same subset of the edges of the subdivision S.

3.2 Three approaches to diagram placement

In the most general case the placement of a square for a region in a subdivision is a problem with two degrees of freedom: the x-coordinate and the y-coordinate of the square to be placed. In the previous subsection we saw that this leads to considering a quadratic number of combinatorially distinct placements. For each of these placements, some linear or quadratic function exists that represents the function to be optimized according to the models *MaxSelfOverlap*, *MinAreaOverlap* and *MinBorderOverlap*. This quadratic function can be determined in $O(n^2)$ time overall by traversing the cells of the arrangement and updating in constant time the quadratic function that was valid in the previous cell. For models *MinMaxOverlapPerc* and *MaxMinUncovered* we need to consider up to m quadratic functions for each cell, which makes the solution less efficient. Using so-called *lower-envelope* computation of quadratic surface patches we can determine the best placements of the squares (Agarwal et al. 1999, Halperin 2004).

In some cases it can be more practical to fix one of the degrees of freedom, say, the y-coordinate, beforehand and then solve the problem optimally over all values of x. Let h be the edge length of the square diagram that we want to place. We choose a y-value y_0. This defines a horizontal strip that goes from y-coordinate y_0 to y_0+h. Graphically speaking, we slide the square from left to right through the strip to determine the best positions. In this case we need to try many different y-values to be sure that the solution we find is nearly as good as the one we get from the algorithm for two degrees of freedom. But if the complexity of the 1d-solution is considerably smaller than that of the 2d-solution sketched above, the 1d-approach may be faster in practice—and it is much easier to implement.

We can go even one step further and fix both the x- and the y-coordinate, and then evaluate the quality of the placement as specified by our models. Then we have to sample many pairs of x- and y-coordinates to get results nearly as good as in the other two approaches.

We refer to these two approaches as 2d-solver, 1d-solver and sampler. In the following subsections we detail how to implement these approaches for the models suggested in Section 2.

3.3 Maximize area of overlap

We first describe the 2d-solver for maximizing the overlap between the diagrams and their regions. We start our description by explaining how we compute the optimal placement of a square Q with respect to an arbitrary simple polygon P.

Consider one of the combinatorially distinct placements of Q, that is, consider the set of edges of P intersecting Q to be fixed. Then Q can still move a little in x- and y-direction. The area of intersection of P and Q changes during such a move, and it is described by a quadratic function in x and y of the form $ax^2+by^2+cxy+dx+ey+f$, where the coefficients a,b,c,d,e,f are constants depending on the edges of P that intersect Q. It follows that in the arrangement in the configuration plane Π, the area of intersection is described by quadratic functions, one for each cell.

To compute which quadratic function applies in each cell, we do a traversal of the arrangement in Π, stepping from one cell to an adjacent cell. If the reference point crosses an edge that separates two cells, the new quadratic function that applies in the new cell can be determined from the previously applicable quadratic function and the coordinates of the edge that was crossed. This takes $O(1)$ time per edge crossed. In every cell we maximize the quadratic function which gives us the coordinates of the reference point (x,y) at its optimal position. We choose the largest of these maxima over all cells to get the placement that maximizes the area of overlap.

This finishes the description of the 2d-solver approach for the maximization of the area of overlap for one square and a simple polygon. For a polygon with n vertices it takes $O(n^2)$ time in the worst case.

We can easily extend the solution to apply to the whole subdivision S. We place a square for a region by only considering the vertices of that region in S. If the m regions have $n_1,\ldots,n_m$ vertices, then the algorithm computes the arrangement for each region separately, in $O(n_i^2)$ time for region R_i. In total we spend $\sum_{i=1}^m O(n_i^2)=O(n^2)$ time because $\sum_{i=1}^m n_i=O(n)$ by Euler's formula.

For the 1d-solver approach, we imagine a horizontal strip whose height is the same as the height of the square Q, and let Q slide horizontally in the strip. The area of intersection of a square and a region R_i changes with a quadratic function in x, the x-coordinate of the reference point of the

square. The quadratic function only changes when the left or right side of the square passes a vertex of R_i. Here we consider the intersections of the strip boundary with the edges of R_i to be vertices of R_i in the strip as well. We will update the quadratic function n_i' times if region R_i has n_i' vertices in the strip. Several different regions may consider the placement of their square somewhere in this strip. If the total number of vertices in the strip is n', then we can determine for all regions the best square placement in the strip in $O(n' \log n')$ time by maintaining all of the at most m quadratic functions in one sweep of the square through the strip.

For a 1d-solver algorithm we must try several different strips to find the best placement for each of the regions. Assume that we try a number of strips from top to bottom, each shifted down by 1/4 of the strip height. Then every vertex of S will be in the interior of at most three strips, or five if we include the boundary of the strip. It is a reasonable assumption that lengths of the edges of S that describe the boundaries of the regions are shorter than the side length of the square to be placed. With this assumption we can bound the number of intersections of strip boundaries and edges of S by $O(n)$ altogether. Now we can bound the total time spent on all strips by $O(n \log n)$. We can also take as the next strip one that is shifted down by $1/h$, for some value $h \geq 1$. In the time bound h will appear as a multiplicative factor.

Note that implicitly, due to the assumptions, we have bounded the number of strips that will be tested by $O(n)$. On can expect that the 2d-solver gives slightly better placements, but the 1d-solver is considerably faster. Since the 1d-solver is very efficient, it is not worthwhile to take the sampler approach, which would take at least linear time.

3.4 Minimize area of overlap with all foreign regions

We next consider the placement of the square Q for a region R_i that minimizes the overlap with all other regions of the subdivision that will be annotated with a square. The center of Q should be inside region R_i for association. Overlap of Q with not annotated regions such as seas does not contribute to the overlap that we minimize. As in the case of maximization of area of overlap described above, we also have a quadratic function for each cell in the arrangement in Π, and we can apply the same solution as before. In this case, the quadratic function gives the overlap with foreign regions, so it must be minimized. Since the center of the square Q must be inside R_i, we need the extended arrangement that captures this condition too, as described at the end of Subsection 3.1. Algorithmically, there is no further difference with the previous case.

Note that it is possible to avoid overlap with seas as well, but with less weight. For example, we could minimize the sum of the areas of overlap with foreign annotated regions and the area of overlap with non-annotated regions divided by 5. In this case the overlap of 0.3 cm^2 with other countries is equally good as overlap of 1.5 cm^2 with seas.

3.5 Minimize length of boundaries covered

We direct our attention to the minimization of overlap with linear features, which can be the boundaries of the subdivision, or additional linear features like highways or rivers. The algorithmic solution for this problem is again very similar to the previous two cases. For ease of description we assume that we only wish to avoid overlap with the boundaries of the input subdivision S, consisting of m regions and n vertices in total. In the same way as in the previous subsections we define and compute a partitioning into combinatorially distinct cells, such that if the square has its reference point in that cell, the length of overlap with the segments of S is described by some simple function. In this case we have a linear function of the form $ax + by + c$ to represent the length of overlap. In the 2d-solver approach, we traverse the partitioning from cell to neighboring cell and update the linear function, eventually leading to the m best placements of the squares, one for each region, in $O(n^2)$ time altogether.

The 1d-solver approach leads to an $O(n \log n)$ time algorithm in the same way as before. The sampler approach is not interesting because the 1d-solver approach is already efficient.

3.6 Minimize foreign region overlap

The models *MinMaxOverlapPerc* and *MaxMinUncovered* both involve a square placement for any region R_i that avoids overlap with other regions, such that it tries to leave as much area as possible to each of the regions. In both cases we must know the area of overlap of the square with *every* other region separately. This implies that for each cell of the partitioning, there are up to m different area-of-intersection functions. This makes these models harder to compute than the previous ones, where only one quadratic or linear function was relevant in a cell. The solutions to *MinMaxOverlapPerc* and *MaxMinUncovered* are essentially the same, algorithmically, and therefore we only consider *MaxMinUncovered* in this subsection.

Assume we want to determine the best placement of a square for region R_i in model *MaxMinUncovered*, and assume the arrangement of combinatorially distinct placements is computed. For any cell γ of the arrangement, the foreign regions intersecting Q when the reference point is in that cell is specified. Only if the square Q intersects a foreign region R_j, the free area of R_j is relevant. It is described by a quadratic function f_j in x and y, similar to the one before. Assume that $R_1, \ldots, R_{m'}$ with $m' \leq m$ is the set of regions that intersect Q if Q's reference point is inside cell γ. For any point $(\hat{x}, \hat{y})$ in γ, the function among $f_1, \ldots, f_{m'}$ that gives the lowest value at $(\hat{x}, \hat{y})$ is the one that determines the smallest uncovered area in the maxmin formula of model *MaxMinUncovered*. If we consider the functions $f_1, \ldots, f_{m'}$ to be mappings from (x, y) to z, the images of the functions are *parabolic surface patches* in 3-dimensional space. The best placement of the reference point of Q inside γ corresponds to the highest point of the surface defined by the pointwise minimum of $f_1, \ldots, f_{m'}$.

We describe an algorithm that computes the best square placement for all regions of the subdivision. First, compute the arrangement of combinatorially distinct placements in the configuration plane. For every cell, determine all relevant quadratic surface patches of regions. Compute the *lower envelope* and the *next level* of the 3-dimensional arrangement of surface patches (why the next level is needed too is described soon). There are algorithms (Agarwal et al. 1999) that compute the lower envelope and next level for all $O(n^2)$ cells of the arrangement in $O(n^2 m^{1+\epsilon})$ time, where ϵ is an arbitrarily small positive constant.

model	2d-solver	1d-solver	sampler
Centroid	$O(n)$	$O(n)$	$O(n)$
MaxSelfOverlap, *MinAreaOverlap*, *MinBorderOverlap*	$O(n^2)$	$O(n \log n)$	$O(n)$
MinMaxOverlapPerc, *MaxMinUncovered*	$O(n^2 m^{1+\epsilon})$	$O(n\alpha(n) \log n)$	$O(n)$

Table 1: Efficiency of square diagram placement.

Next we go through all cells of the arrangement again. For any cell γ, only one region R_i can accept a square whose reference point is inside γ, because the center of Q must be inside the region it annotates. Region R_i also has its free-area function f_i which can be involved in parts of the lower envelope. This is in fact quite likely, especially if R_i is small. Let f be the function that projects each point in γ to the lower envelope if f_i is above the lower envelope, and to the next level otherwise. The maximum of the function f gives us the best placement in cell γ. After computing the best placements for all cells of the arrangement, we can select the best placement for every region R_i by picking the best one from all cells that would annotate R_i. The computation of these placements, once the lower envelope and next level are known, can be done in time linear in the complexity of the lower envelope and the next level. The total time spent is $O(n^2 m^{1+\epsilon})$.

The 2d-solver approach is not so efficient and rather complex to implement. We therefore also consider the 1d-solver approach. We consider a horizontal line on which the reference point of the square moves, and have at most m quadratic functions in x only. We compute the lower envelope and the next level of the arrangement of quadratic functions in $O(n'\alpha(n') \log n')$ time (Hershberger 1989), where n' is the number of vertices of S in the strip and $\alpha(n')$ is the extremely slowly growing functional inverse of Ackermann's function.

With the same two assumptions as in Subsection 3.3, namely (i) every segment in S is no longer than the side length of the square, and (ii) the next strip considered is shifted down over a distance $1/h$ for an integer constant $h \geq 1$, we can prove that the total running time is $O(n\alpha(n) \log n)$.

3.7 Efficiency of the algorithms

Table 1 summarizes the asymptotic time bounds for placing all m square diagrams with their regions $R_1, \ldots, R_m$ according to aour six models. Recall that n is the total complexity of the subdivision, m is the number of regions, $\alpha(n)$ is the extremely slowly growing functional inverse of Ackermann's function, and ϵ is a positive real constant that may be chosen arbitrarily small. The time bounds for the 1d-solver and sampler approaches depend on certain realistic assumptions of the input. In particular, we assume that the edges of the subdivision S are shorter than the side length of the squares to be placed (or, at worst, a constant times as long). Secondly, we assume that in the 1d-solver approach, the separation of y-coordinates chosen are separated by at least some fixed fraction of the square height, like 1/10. Both assumptions can be expected to hold in practice.

Table 1 shows that the 1d-solver is very efficient in all cases. The 2d-solver is less efficient, but can still be used when computation time is not critical. For models *MinMaxOverlapPerc* and *MaxMinUncovered* the 2d-solver approach has the disadvantage of requiring considerable implementation efforts.

3.8 Circular and different-size diagrams

There are two extensions to fixed-size square diagram placement, namely, circular diagram placement and diagram placement for different size diagrams. The main complication of circular diagram placement is that the function that represents the area of intersection is no longer quadratic. Instead, it is a function with terms that involve square roots. The number of terms is linear in the number of edges intersected, and an optimal placement cannot be computed analytically anymore even for a single cell of the arrangement. This problem already arises in the 1d-solver approach. Either the optimum for a cell must be determined by algorithms that approximate the roots of analytic functions, or the sampler approach can be used (which, of course, is also an approximation strategy). The sampler approach for fixed-size circles has the same performance as for squares under the same assumptions (see Table 1).

Next we consider the situation that the regions $R_1, \ldots, R_m$ are to be annotated by square diagrams of different sizes. The main complication is that we cannot use the arrangement from Subsection 3.1 anymore, because it is defined for one pre-specified square size. Of course, we could consider m different arrangements, one for each of the square sizes specified. But this would make the time bounds for all 2d-solver algorithms increase by a factor of m. For the 1d-solver and sampler approach, the increase in running time depends on the adaptations of the assumptions we used; considerably stronger assumptions are needed to prove running times better than approximately $O(nm)$.

In practice, the 1d-solver approach can benefit from the fact that horizontal strips of a certain height are only needed in the vicinity of regions that require squares of that size. This will reduce the total number of strips considered considerably. In the worst case, however, $\Theta(mn)$ strips are considered.

4. IMPLEMENTATION AND EXPERIMENTS

We have implemented the sampler approach described in Section 3.2 for all models presented here in C++, using the CGAL library and the g++ compiler on a 2GHz AMD Athlon under Linux. The purpose of the implementation is to find out which of the models gives the best placement of diagrams on maps. For this purpose the sampler approach with a small step size is sufficient to compare the models and obtain a good indication of how well they perform.

We used CGAL Nef polyhedra to compute the area of overlap between a country and a label. Unfortunately, it turned out that computing intersections between Nef polyhedra is very slow, so we have implemented the boundary

map	model / label size	Centroid #	Centroid %	MaxSelf-Overlap #	MaxSelf-Overlap %	MinArea-Overlap #	MinArea-Overlap %	MinBorder-Overlap #	MinBorder-Overlap %	MinMax-OverlapPerc #	MinMax-OverlapPerc %	MaxMin-Uncovered #	MaxMin-Uncovered %
Europe	40	3	1.96	6	0.25	0	0	5	0.16	1	0.01	1	0.14
	50	8	2.59	10	0.99	1	0.35	7	0.43	3	0.37	5	0.41
	70	23	5.81	24	3.15	11	0.73	13	1.81	8	0.80	15	3.31
USA	40	5	0.63	1	0.00_5	1	0.01	8	0.36	1	0.01	2	0.05
	50	13	1.28	10	0.25	4	0.06	13	4.24	4	0.05	5	0.20
	70	31	4.33	27	2.16	18	1.17	34	15.33	21	1.35	22	3.38
Mexico	40	4	0.25	2	0.12	1	0.00_4	2	0.11	0	0	4	0.28
	50	9	0.74	6	0.35	4	0.05	4	2.56	1	0.00_3	6	1.54
	70	21	3.86	18	2.32	8	0.51	12	8.14	8	0.90	9	2.37

Table 2: Number of label–label overlaps (#) and percentage (%) of label area overlapped by other labels.

overlap without using these structures. The implementation of the *Centroid* model was largely inspired by O'Rourke (1995a), which is a solution of an exercise in the book by O'Rourke (1995b). The implementation is not optimized, except that it only computes the intersection of a label and a country if their bounding boxes overlap. This simple idea decreased the running time by a factor of about 10.

We have tested our implementation on maps of Europe (Figures 3–8 and 11), the United States (Figures 9 and 12) and Mexico (Figure 10). The maps of the U.S. and Mexico map stem from a library which is part of xfig, a freeware drawing program. These maps consist of 1638 and 4718 vertices, respectively. The map of Europe consists of 2533 vertices. We obtained it by digitizing a map available in the Internet, since the map provided in the xfig library is not up-to-date.

We labeled each country or state with a label whose area is 70% of the area of the labeled region, which is probably larger than the average size of commonly found diagrams. We chose this size since it is a good tradeoff—large enough to show significant differences between various models, but small enough to preserve the overall legibility of the map. The rectangular labels have an aspect ratio of 1.2, see Figures 3–10. We have implemented the pie-chart placement by approximating a circle by a 20-gon and then placing a circular label at the position calculated for the 20-gon, see Figures 11 and 12. In case several label candidates had the same value of the target function, we chose the one whose area of overlap with the labeled country is largest (*MaxSelfOverlap*). The other candidates are depicted in light grey.

5. RESULTS

Labeling the map of Europe according to the different models, see Figures 3–8, took 1, 145, 297, 20, 297, and 297 seconds, respectively. Labeling the map of the U.S. and Mexico according to *MinMaxOverlapPerc*, see Figures 9 and 10, took 55 and 171 seconds, respectively. These running times were almost independent of the label sizes. The differences result from the use of Nef polyhedra except for models *Centroid* and *MinBoundaryOverlap*, from the differences of precision in the sampling, and from the fact that for model *MaxSelfOverlap* one only has to compute the overlap with the labeled country. Placing pie charts took somewhat longer, namely 695 and 173 seconds for the U.S. map in Figure 11 and the map of Mexico in Figure 12, respectively.

For the three maps of Europe, the U.S., and Mexico, and for each of the six models, Table 2 lists the number of label–label overlaps and the percentage of the label area that is overlapped by other labels. These experiments were done with rectangular labels of aspect ratio 1.2 and an area of 40, 50, and 70% of the region's area. Since the corresponding table for pie-chart placement looked very similar, we have dropped it.

First of all, as expected the simpler models (*Centroid* and *MaxSelfOverlap*) cause a lot of label–label intersection. For model *MaxSelfOverlap* (see Figure 4), there is no difference between intersecting a sea or intersecting another country, so this model does not take advantage of the proximity of oceans.

A second observation is that the area models tend to give better results than boundary models. Model *MinBorderOverlap* is the only boundary model presented here, but we have tested many others, and none was comparable to the area models for our sample maps. Nevertheless, some other maps, with rivers or road networks, may take advantage of a boundary model, where the boundary length would include the length of overlapped rivers and roads.

Of those area models that take other countries into account, the worst seems to be *MaxMinUncovered* (Figure 8). The disadvantage of this model is that it tends to *completely* avoid the overlap with the smallest neighboring countries, and so the labels of big neighbor countries tend to overlap each other.

The differences between the models *MinMaxOverlapPerc* and *MinAreaOverlap* are not very significant on the maps we tested, the former performing slightly better than the latter. However, this difference may grow if many small countries are far from oceans, since *MinMaxOverlapPerc* avoids overlapping large parts of countries, and so tends to prevent overlapping small countries while on the other hand not being as strict as *MaxMinUncovered*.

In our opinion, the best model is model *MinMaxOverlapPerc*. Another good choice is model *MinAreaOverlap*, which yields good results in most cases and has the advantage that a 2d-solver for this model is simpler and asymptotically faster than for *MinMaxOverlapPerc*.

6. CONCLUSIONS

We have studied the problem of placing square and circular diagrams on the regions of a map under various criteria. The main objective was to avoid overlap of different dia-

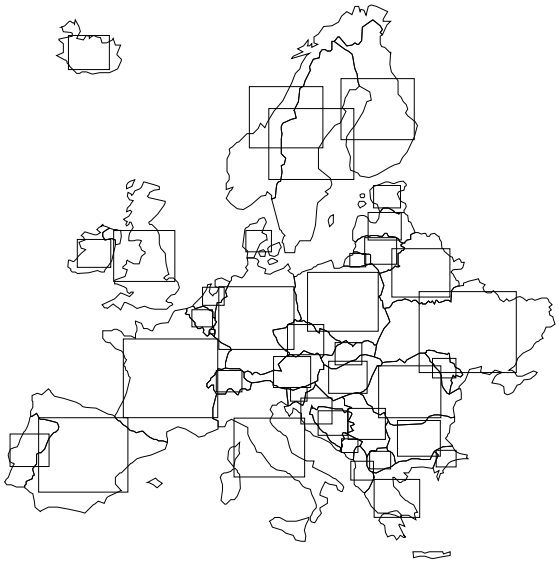

Figure 3: Model *Centroid*

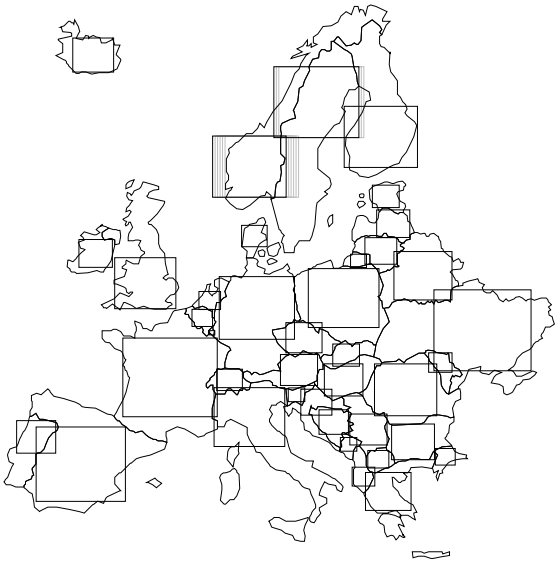

Figure 4: Model *MaxSelfOverlap*

grams, without resorting to optimization techniques like hill climbing or simulated annealing. Of course it is possible to use the placements that we computed as a starting point for such an optimization. We presented three algorithmic approaches to diagram placement and discussed their efficiency. The results also apply directly to rectangular diagrams of fixed aspect ratio.

Of the different criteria, the best ones appears to be (i) minimizing the maximum of the percentages of overlap with other regions, and (ii) minimizing the area of overlap with other regions. Experiments show that the diagrams are placed well in most cases for these criteria, and area of overlap among diagrams is minor. Which of the two methods is usually better requires further testing.

ACKNOWLEDGMENT

The authors thank Danny Halperin for information and references on upper envelopes of surface patches.

REFERENCES

Agarwal, P. K., Efrat, A., and Sharir, M. (1999). Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM J. Comput.*, 29:912–953.

Arikawa, M. and Kambayashi, Y. (1991). Dynamic name placement functions for interactive map systems. *The Australian Computer Journal*, 23(4):133–147.

Bentley, J. L. and Ottmann, T. A. (1979). Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28(9):643–647.

Chazelle, B. and Edelsbrunner, H. (1992). An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39(1):1–54.

de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (2000). *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2nd edition.

Dent, B. D. (1999). *Cartography: Thematic Map Design*. McGraw-Hill, 5th edition.

Edmondson, S., Christensen, J., Marks, J., and Shieber, S. (1997). A general cartographic labeling algorithm. *Cartographica*, 33(4):13–23.

Glassman, B., editor (1996). *The Macmillan Visual Almanac*. Macmillan, New York.

Halperin, D. (2004). Arrangements. In Goodman, J. and O'Rourke, J., editors, *Handbook of Discrete and Computational Geometry*, chapter 24, pages 529–562. Chapman & Hall/CRC, Boca Raton, 2nd edition.

Harrie, L., Stigmar, H., Koivula, T., and Lehto, L. (2004). An algorithm for icon placement on a real-time map. In *Proc. 11th Int. Symp. on Spatial Data Handling*.

Hershberger, J. (1989). Finding the upper envelope of n line segments in $O(n \log n)$ time. *Inform. Process. Lett.*, 33:169–174.

Mulmuley, K. (1993). *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, Englewood Cliffs, NJ.

O'Rourke, J. (1995a). centroid.c. Available at ftp://cs.smith.edu/pub/code/centroid.c.

O'Rourke, J. (1995b). *Computational Geometry in C*. Cambridge University Press, 2nd edition.

Pinto, I. and Freeman, H. (1996). The feedback approach to cartographic areal text placement. In Perner, P., Wang, P., and Rosenfeld, A., editors, *Advances in Structural and Syntactical Pattern Recognition*, pages 341–350. Springer-Verlag, New York.

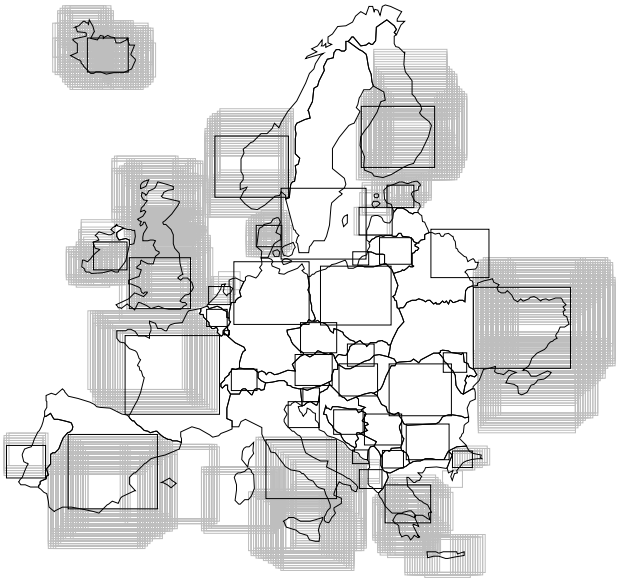

Figure 5: Model *MinAreaOverlap*

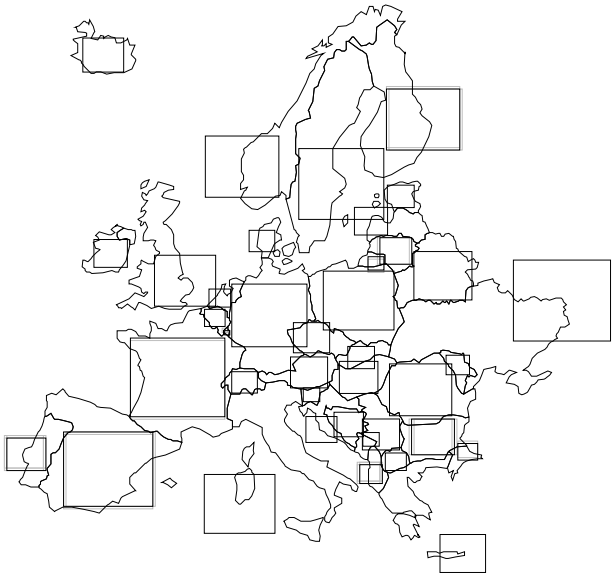

Figure 6: Model *MinBorderOverlap*

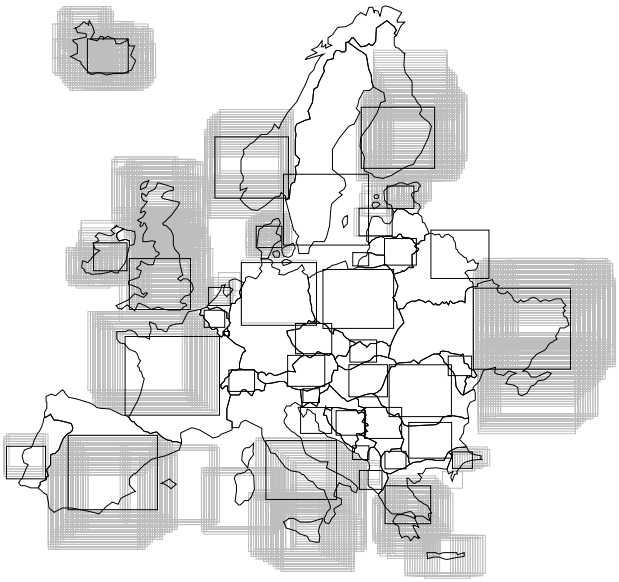

Figure 7: Model *MinMaxOverlapPerc*

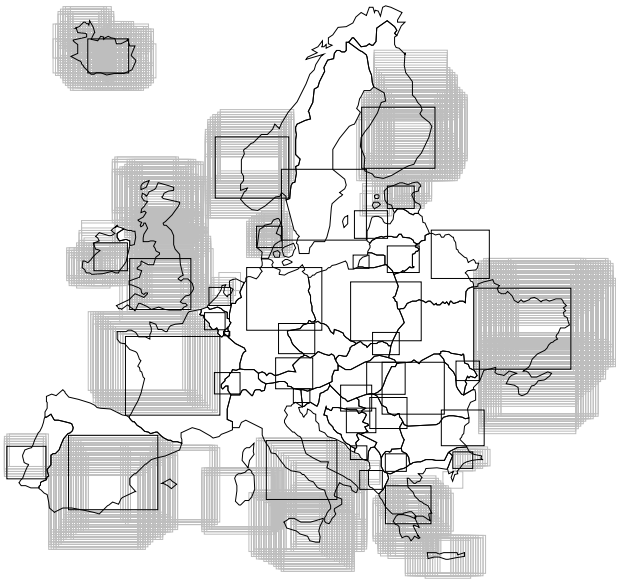

Figure 8: Model *MaxMinUncovered*

Robinson, A., Morrison, J., Muehrcke, P., Kimerling, A., and Guptill, S. (1995). *Elements of Cartography*. John Wiley & Sons, New York, 6th edition.

Tóth, G. F. (2004). Packing and covering. In Goodman, J. and O'Rourke, J., editors, *Handbook of Discrete and Computational Geometry*, chapter 2, pages 25–52. Chapman & Hall/CRC, Boca Raton, 2nd edition.

van Roessel, J. W. (1989). An algorithm for locating candidate labeling boxes within a polygon. *The American Cartographer*, 16(3):201–209.

Wolff, A. and Strijk, T. (1996). The Map-Labeling Bibliography. `http://i11www.ira.uka.de/map-labeling/bibliography/`.

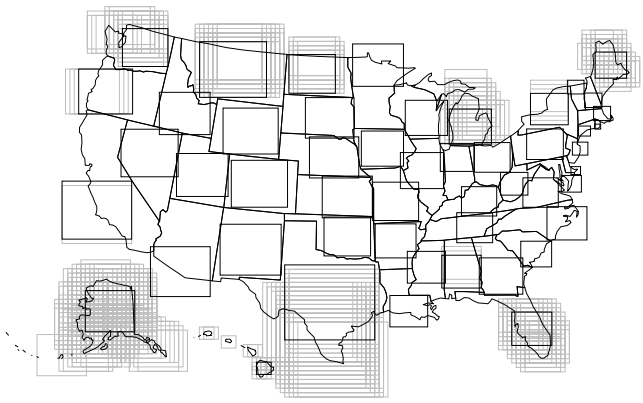

Figure 9: USA, model *MinMaxOverlapPerc*.

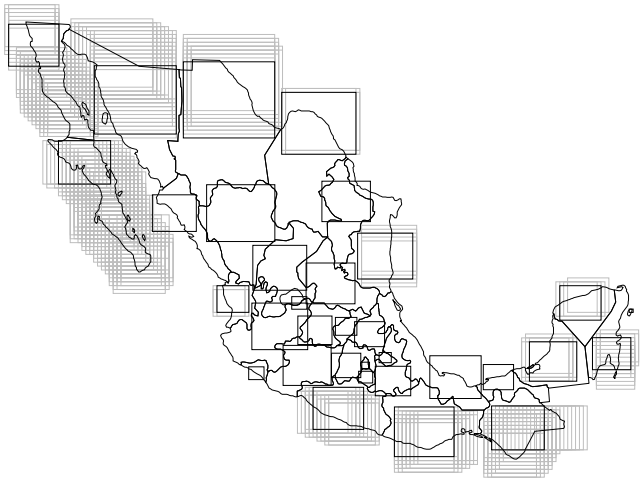

Figure 10: Mexico, model *MinMaxOverlapPerc*.

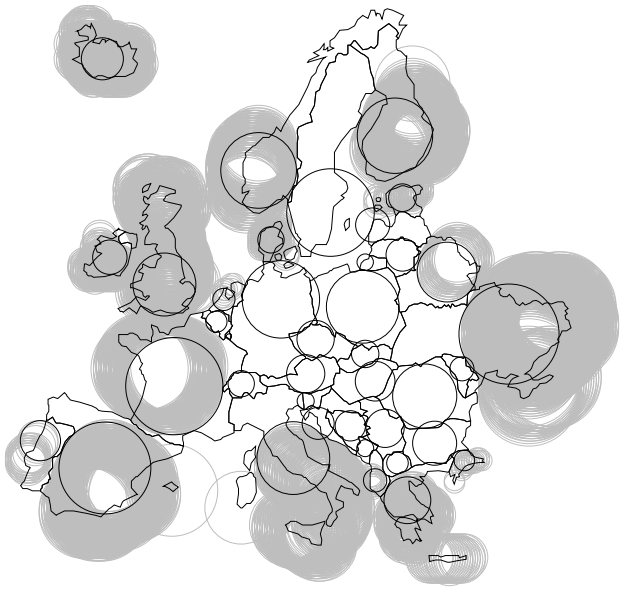

Figure 11: Europe, model *MinMaxOverlapPerc*.

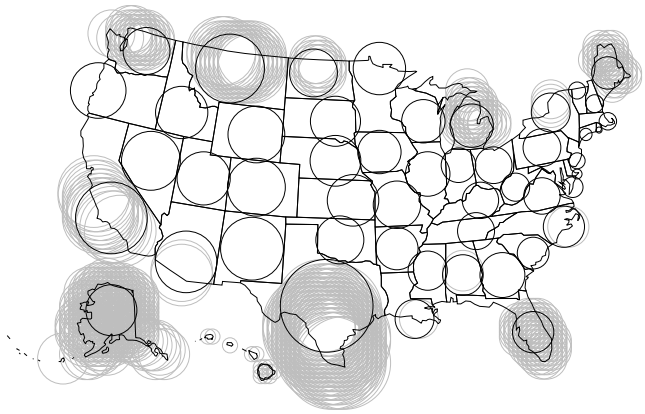

Figure 12: USA, model *MinMaxOverlapPerc*.