A General Cartographic Labeling Algorithm

Shawn Edmondson
Baker Hill Corp.

Jon Christensen
Painted Word Inc.

Joe Marks
MERL – A Mitsubishi Electric Research Laboratory

Stuart M. Shieber
Harvard University

December 19, 1996

Abstract

Some apparently powerful algorithms for automatic label placement on maps use heuristics that capture considerable cartographic expertise but are hampered by provably inefficient methods of search and optimization. On the other hand, no approach to label placement that is based on an efficient optimization technique has been applied to the production of general cartographic maps — those with labeled point, line, and area features — and shown to generate labelings of acceptable quality. We present an algorithm for label placement that achieves the twin goals of practical efficiency and high labeling quality by combining simple cartographic heuristics with effective stochastic optimization techniques.

To appear in *Cartographica*.

1 Introduction

Many apparently compelling techniques for automatic label placement use sophisticated heuristics for capturing cartographic knowledge, but, as noted by Zoraster (1991), also use inferior optimization strategies for finding good tradeoffs between the variety of competing concerns involved in typical labeling problems. These techniques use procedural methods or "if-then" production rules to represent cartographic knowledge about good label-placement practice, and a variant of depth-first search to explore different labelings (Doerschler and Freeman, 1992; Ebinger and Goulette, 1990; Jones, 1989).

However, depth-first search is now known to be a markedly inferior technique for finding near-optimal labelings in the set of all possible labelings. Some of the more powerful optimization strategies that have been proposed for label placement include:

- *physical relaxation* – while remaining tethered to the symbols they tag, labels are moved smoothly in response to virtual forces generated by label-label and label-symbol overlaps (Feigenbaum, 1994; Hirsch, 1982);
- *zero-one integer programming* – scores associated with each label's candidate positions (a discrete set) are refined iteratively to better reflect the relative desirability of the different positions (Zoraster, 1986; Zoraster, 1990);

- *gradient descent* – a randomly generated labeling is improved monotonically by considering all alternative positions for each label (again chosen from a discrete set) and making the single label move that most improves the quality of the whole labeling (Christensen, Marks, and Shieber, 1993); and
- *simulated annealing* – a generalization of gradient descent in which single label moves that worsen the quality of the labeling are performed occasionally in the hope of avoiding bad labelings that happen to be locally optimal (Christensen, Marks, and Shieber, 1993).

These techniques all outperform depth-first search by a wide margin (Christensen, Marks, and Shieber, 1995).

So why have these supposedly superior techniques not been adopted widely? In general, implementations of the better optimization strategies have not dealt with the full range of cartographic features (that is, point, line, and area features), nor have they incorporated sufficient cartographic knowledge about text placement. For these reasons, the maps they produce have not been persuasive. However, this shortcoming is not intrinsic, at least for some of the aforementioned techniques. In this paper we show how detailed cartographic knowledge and powerful optimization can be combined effectively. The resulting algorithm is general (it can label point, line, and area features), efficient (it can label dense, page-sized maps in seconds on a computer workstation), and effective (the resulting labelings are visually appealing and consistent with good cartographic practice). Furthermore, compared to existing algorithms, it is concisely stated and easily implemented.

The key to our approach is a careful separation of the cartographic knowledge needed to recognize a good labeling from the optimization procedure required to find one. Our notion of cartographic knowledge is a procedure for computing an absolute numeric score that is properly indicative of a labeling's quality; no other properties of this scoring function (such as continuity, differentiability, or definability as production rules) are assumed. We treat the scoring function as an arbitrary objective function to be optimized, and apply a powerful optimization procedure to find its near-optimal values. These values correspond to near-optimal labelings.

To establish the viability of this concept, we describe a particular optimization procedure and a particular scoring function that can be used together in the framework we propose above. Neither are necessarily optimal: it was not our intent to provide the ultimate solution for automatic map lettering, a long-term goal that is probably best left to professional cartographers, not computer scientists. However, we believe that we have identified the framework within which the ultimate solution probably lies. As evidence for this claim, we present a simple system prototype that generates visually compelling labelings for dense page-sized maps in under ten seconds on a personal computer.

We begin by describing our technical approach in sufficient detail to enable our work to be replicated in its entirety (Sections 2 through 5). We then present some sample maps labeled using an implementation of our method (Section 6), and conclude with an analysis of present and future work on label placement (Section 7).

2 Technical Approach

Our approach is predicated on a division of the label-placement task into three essentially independent subtasks. They are:

1. *Candidate-position generation:* Given a point, line, or area feature, identify a set of candidate locations for its label. A *labeling* is then a set of label positions, one drawn from each feature's set of candidate positions.

2. *Position evaluation:* Given a labeling, efficiently compute a score that indicates its quality with respect both to the position of labels relative to the tagged symbology, and to spatial contention between the label and other features and feature labels.

3. *Position selection:* Given a set of candidate label positions for each map feature, choose one label position from each set so that the overall quality of the labeling, as determined by the evaluation method, is as high as possible.

We discuss these subtasks in reverse order in the next three sections. Because of the simplicity of the position-selection method, we discuss it briefly; full details have been presented elsewhere (Christensen, Marks, and Shieber, 1995). We then consider the problems of label-position evaluation and generation, which form the core of our contribution in this paper.

3 Position Selection

Given a set of generated candidate positions for each label and an overall evaluation function, selecting positions for all the labels so that the evaluation function is globally minimized[1] is an optimization task. Although many different methods have been proposed for this task, we favor a method based on *simulated annealing* (Kirkpatrick, Gelatt Jr., and Vecchi, 1983):

1. For each feature, place its label randomly in any one of the candidate positions for that feature.

2. Initialize a "temperature" T to an initial high value.

3. Repeat the following steps until the rate of improvement falls below a given threshold:

   (a) Decrease T according to an annealing schedule.

   (b) Pick a feature randomly and move its label to a new position randomly chosen from that feature's set of candidate positions.

   (c) Compute ΔE, the change in the overall labeling's evaluation caused by repositioning the label.

   (d) If the new labeling is worse, undo the label repositioning with probability $P = 1.0 - \exp(-\Delta E/T)$.

The algorithm is almost completely specified in the outline above. The remaining details concern the initial temperature (chosen so that $P = 2/3$ when $\Delta E = 1$), the annealing schedule (periodically the temperature decreases by 10 percent), and the termination condition (evaluation of $5n$ consecutive repositionings with no changes made, where n is the number of labeled features). Full details concerning these issues are provided elsewhere (Christensen, Marks, and Shieber, 1993; Christensen, Marks, and Shieber, 1995). It suffices here to note that the parameters are such that 100,000 computations of ΔE are typically required for convergence on problems involving up to 1,500 labeled features. Thus the overall efficiency of the label-placement algorithm depends crucially on efficient computation of ΔE.

This optimization method has several advantages. First, it is effective at finding near-optimal solutions to label-placement problems. Christensen, Marks, and Shieber (1995) demonstrate through

[1] Our statement of the evaluation function associates lower values with better labelings, so that the optimization problem is one of minimization. This choice is, of course, arbitrary.

exhaustive empirical testing of point-feature label-placement algorithms that simulated annealing dominates all previously reported practical algorithms for this problem. The results were robust over both actual cartographic data and randomly generated data, and thus provide some foundation for the use of artificial data in the present work. Second, the method is efficient; labelings are generated in an amount of time that is reasonable and competitive with other algorithms.[2] Finally, it makes no assumptions about the evaluation function beyond the efficient computability of the numeric difference ΔE. This last advantage suggests that the method should generalize beyond the point-feature label-placement problem to which it was initially applied. The remainder of this paper can be seen as a verification of this suggestion.

4 Position Evaluation

We now turn to the task of evaluating label positions. In order to compare various labeling solutions, we require the ability to compute a single numeric score, E, that indicates the quality of a labeling. Furthermore, the change in score, ΔE, that results from repositioning a single label must be efficiently computable. In our algorithm, each label's contribution to E is computed as a weighted sum of simple *metrics*, which are described below.

For each metric, we define an *ideal case* and a *borderline case*. In the ideal case, the candidate position is ideally located as far as the given metric is concerned; in the borderline case, the position is poor, but barely acceptable. Metrics will be designed to yield a value of 0.0 for ideal cases, 1.0 for borderline cases, and higher values for objectionable cases.

First, we consider metrics that quantify spatial crowding and overlap. Next, we examine positioning metrics that depend on the spatial relationship between a label and the feature it tags. We conclude with a description of how the various metrics are combined into a single evaluation function.

4.1 Overlap metrics

Feature overlap. In order to discourage label-symbol overlaps, we need to know the number of feature symbols (the actual circles, polylines, and polygons comprising a map's nontextual symbology) that a given label overlaps. We consider three metrics, *PointOver*, *LineOver*, and *AreaOver*, which measure overlaps with point, line, and area symbols, respectively. When there are no overlaps of a given kind, we have the ideal case. We will take the borderline case to be a single overlap. Thus, one reasonable metric is to simply count the number of overlaps. This is exactly what is done for the metric *PointOver*, which is defined as the number of overlaps between a selected label position and all point-feature symbols.

For area and line features, however, we need a more subtle metric. A label position that intersects and is parallel to the polyline of a line feature or an area border is completely unacceptable. However, a position that intersects the polyline at right angles might qualify as barely acceptable. We define *LineOver* and *AreaOver* accordingly: Let p_1 and p_2 be the points where a polyline enters and exits the label's bounding rectangle. (See Figure 1.) Let the vector $\hat{b}$ point in the direction of the label's baseline (east in Figure 1) and the vector $\hat{v} = (p_2 - p_1)/|p_2 - p_1|$ approximate the direc-

[2]Both of these findings are remarkable. Simulated annealing is usually thought of as a very inefficient algorithm of "first resort" that is used only until a better approach is devised. However, for point-feature label placement (the purest form of the problem, and the most useful for comparing algorithms) it takes only a few seconds to label dense maps with hundreds of point features, and it produces better labelings than any algorithm developed during the 15 years or so for which automatic label placement has been studied seriously (Christensen, Marks, and Shieber, 1995).

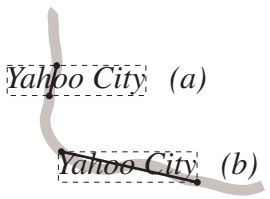

Figure 1: Calculation of $\hat{v}$ for the overlap metric *LineOver*.

tion of the polyline where it intersects the label. The value of the metric (*AreaOver* or *LineOver*) is then $1 + 9|\hat{v} \cdot \hat{b}|$.

This metric can have value 0.0 (no intersection between label and line: ideal), or can range from 1.0 (label and line intersect and are perpendicular: barely acceptable) to 10.0 (label and line intersect and are parallel: unacceptable). In the example of Figure 1, *LineOver* is 2.56 for (a), and 9.86 for (b). These values are appropriate, since (a) is a much better placement than (b). Of course, neither is as good as a placement that does not intersect the line at all, which would yield zero for *LineOver*.

AreaOver and *LineOver* discard much detail — everything but the direction of the polyline and the label. However, they can be computed very efficiently. In experiments, a more elegant metric based on the area of intersection between the label and the polyline proved to do a slightly better job of evaluating label-line intersections, but at a high cost in efficiency.

Label overlap. Mutually overlapping labels are clearly undesirable. As with the *PointOver* metric, we simply count the number of overlaps: we define *LabelOver* to be the number of overlaps between one selected label position and all others.

It is possible to precompute values for almost all of our metrics prior to position selection. For example, *LineOver* can be precomputed for each candidate position without any knowledge of actual selected label positions. The exception to this rule is *LabelOver*, which must be recomputed continually during position selection. Fortunately, we are able to accomplish this reevaluation quickly by precomputing and storing all pairs of intersections between candidate label positions.

PointOver, *LineOver*, *AreaOver*, and *LabelOver* together comprise all the metrics concerned with spatial contention among labels and symbology. For simplicity, we have not included metrics for other types of label-symbology overlap, e.g., mountain ranges, vegetation, etc. However, the introduction of additional overlap metrics is straightforward. We now turn to metrics that govern the positioning of a label with respect to the feature that it tags.

4.2 Point-positioning metrics

The sole metric for point-feature labels, *PointPos*, is determined by a straightforward ranking of a discrete set of 17 candidate positions surrounding a point. The actual values used are illustrated in Figure 2, and described more fully in Section 5.1. These values are based loosely on Imhof's well-known guidelines (Imhof, 1962):

- Label positions to the right of a point are preferred to those on the left.
- Labels above a point are preferred to those below.

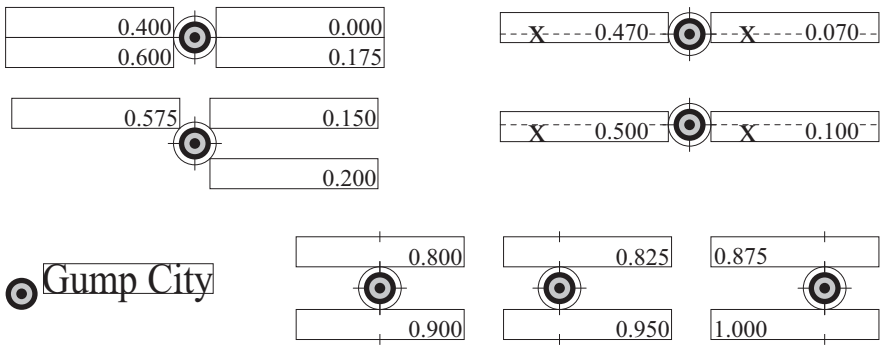

Figure 2: An example of 17 candidate label positions for a point feature and their *PointPos* values.

- The more a label's baseline is offset from a horizontal line through the center of its associated point, the less favored it is.

Finally, the three positions above the point (with penalties 0.8, 0.825, and 0.875) may be shifted up slightly if a descender would otherwise overlap the point feature. In that case, an additional penalty of 0.25 is added.

4.3 Line-positioning metrics

Before describing the various line metrics that figure in the evaluation function, we must first define some useful terms and measures, illustrated in Figure 3. A line feature is represented as a long, thin polygon. However, we refer to such a polygon as a "polyline" throughout, since any computation is done with respect to the actual polyline that forms the nearest side of the line feature's polygon (whether above or below the label). The *baseline* of a label position is the line upon which the characters are drawn. However, we also take into account the shape of the text, defining the *skyline* of a label to be the union of the bounding boxes of the label's characters. The skyline extends both above and below the baseline. The *ideal distance* δ from the baseline of a label to a perfectly straight line feature below it is included in several of the line-metric formulas. The ideal distance should vary with both the thickness of the line feature and the type size, so we set $\delta = \mathit{ascent}/4 + \mathit{thickness}/2$, where the ascent is the distance from the baseline to the top of capital letters of the label. The *swath* is an infinitely long strip which is perpendicular to the baseline and centered about the label. The width of the swath is 20% greater than the width of the label. The part of the polyline near the label that intersects the swath is termed the *swath line*.

We include five positioning metrics for line-feature labels in our evaluation function. The first three measure the label's relationship to the swath line: *AveDist* and *MinDist* measure the average and minimum distance from the label to the swath line; *Flatness* measures the degree of curvature of the swath line. The final two metrics measure aspects of the relationship between the label and the line feature as a whole. *Centeredness* looks at the proximity of the label to the center, measured end-to-end, of the line feature. *Aboveness* simply indicates whether the label is above or below the line feature. We discuss these metrics in more detail below.

In the following discussion we will assume we are measuring a label that is above the line feature. The rules and methods work the same if the label happens to be below the line; we simply use the other side of the skyline.

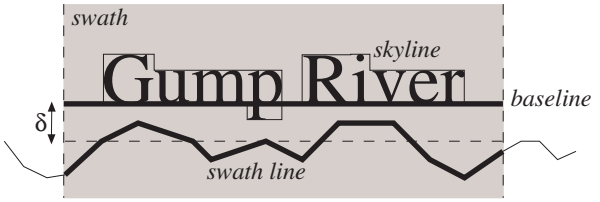

Figure 3: Terminology for line-feature labels. The *swath* is bounded on the left and right by the vertical dashed lines.

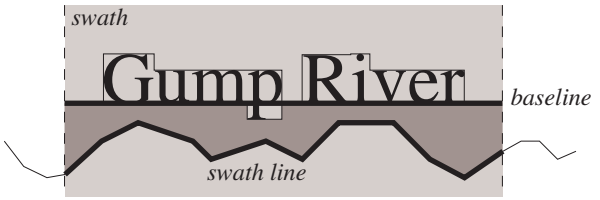

Figure 4: Computing average distance.

Average distance. Positioning a label at an appropriate distance from its line feature is crucial. To compute a local measure of the average distance d from a label to its associated line feature, we take the area between the swath line and the lower side of the skyline, then divide by the width of the swath. The relevant area is shaded darkly in Figure 4.

Next we incorporate the average distance d into a useful metric with the following characteristics. First the metric should yield the optimal measure of 0.0 for an average distance δ, the ideal distance defined above. Next, let the borderline case be an average distance of either 0.0 or 2δ. We choose to let *AveDist* grow as the square of the deviation from δ, since experiment has shown this to work better than a simple linear model. Thus, we arrive at $AveDist = (d - \delta)^2/\delta^2$. By way of example, Figure 5 shows 10 label positions with low *AveDist* values for a given line feature. The position drawn in bold has the best value of the 10.

Minimum distance. *AveDist* measures the average distance from a label to its line feature in the label's neighborhood. However, due to curvature of the line or descenders in the text, some feature

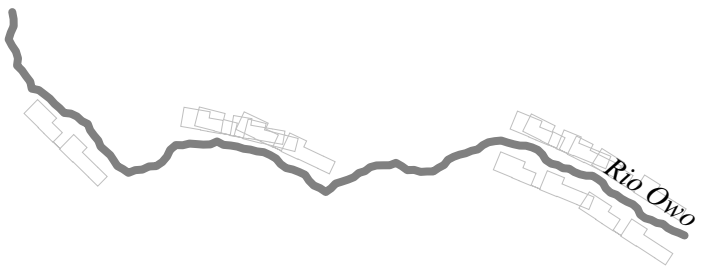

Figure 5: Label positions with good *AveDist* values.

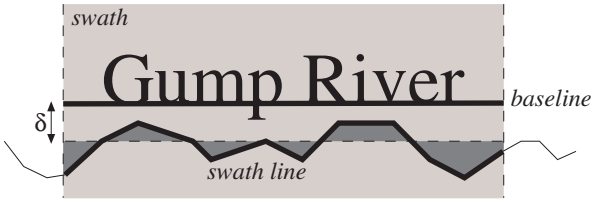

Figure 6: Computing flatness.

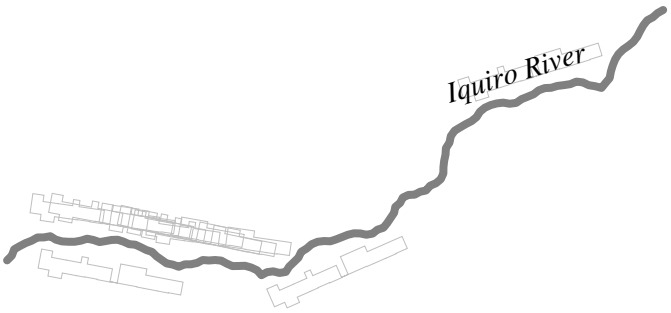

Figure 7: Label positions with good *Flatness* values.

segments might lie very close to the label, or actually on it, without making *AveDist* large. We need a metric to quantify this potential anomaly. A reasonable measure is the minimum distance d' between any two points p_1 on the swath line and p_2 on the skyline. Ideally, d' is the ideal distance δ; barely acceptable values are $d' = 0$ and $d' = 2\delta$. A function that obtains these values is $MinDist = (d' - \delta)^2/\delta^2$. However, as we will see in Section 5, we do not need to consider the *MinDist* metric explicitly when evaluating a labeling, because our generation algorithm only creates candidate positions with perfect *MinDist* values.

Flatness. It is better to locate labels where the associated line feature is relatively flat. To quantify flatness, we compute the deviation of the swath line from a straight line L that is parallel to the baseline and offset the ideal distance from it. We sum up the area between the swath line and L, then divide by the width of the swath. The area sum is shaded darkly in Figure 6.

Let the quotient — representing the swath line's deviation from a straight line — be d''. In the ideal case, d'' is zero; we will pick the borderline case to be when $d'' = \delta$. As with *AveDist* and *AreaPos*, we let *Flatness* grow as the square of d''. Hence, $Flatness = d''^2/\delta^2$. Several label positions with low *Flatness* values for a sample line feature are shown in Figure 7. The lowest-valued position is drawn in bold.

Aboveness. Consider a horizontal line feature and a label running left to right: the label may be placed above or below the line. Following Imhof, we prefer labels above the line to those below. Therefore the metric *Aboveness* has a value of 0.0 if the position is above, 1.0 if below.

Centeredness. Currently, our heuristics assume that each line feature will have only one label. Therefore it is important that labels lie near the centers of their associated lines. To evaluate this

quality, we need a global metric *Centeredness*, which we compute as follows. We find the point p on the polyline closest to the midpoint of the label baseline. Let l_1 be the distance *along the polyline* from one end to p; let l_2 be the polyline's total length. Then $l = l_1/l_2$ ranges from 0.0, if p lies at an end, to 1/2, if p lies in the middle, to 1.0, if p lies at the other end. The ideal case is when $l = 1/2$; the borderline case is when l is 0.0 or 1.0. A suitable metric is thus $Centeredness = |2l - 1|$.

4.4 Area-positioning metrics

Area-specific metrics quantify the relationship between a label position and its area feature. We will assume that area features are usually large enough to accommodate their labels, which consist of closely spaced horizontal text. Given this assumption, a single, simple metric is generally all that is required for satisfactory area-feature labeling. The metric, *AreaPos*, measures the proximity of a label to its area's centroid.[3] Let c be the distance from the center of a label position to the centroid of its area feature. In the ideal case, $c = 0$. Let s be the distance from the area's centroid to its furthest vertex. We will take the borderline case to be when $c = s$. A suitable linear function is therefore $AreaPos = c/s$.

4.5 The overall evaluation function

The overall evaluation function is a weighted sum of the metrics described above for each label on the map. Suitable values for the weights were set intuitively and refined empirically; they are summarized in Table 1. Notice that the weight given to the *MinDist* metric is irrelevant, since the position-generation procedure for line-feature labels (Section 5) is guaranteed to generate only positions for which the *MinDist* value is perfect (0.0).

5 Position Generation

While we express most of the system's cartographic knowledge in the evaluation function, it is clearly helpful for the generation routines to be created with some knowledge of the evaluation metrics. We have identified three qualities of a good position-generation method:

1. For efficiency during position selection, the number of generated positions should be relatively small. For example, in our system, the generation algorithm never provides more than 32 candidate positions for each feature.

2. The generation method should strike an appropriate balance between efficiency and quality: identifying only high-quality positions is too expensive and difficult, whereas naive methods tend to produce too many low-quality candidates. An appropriate balance can be achieved by keeping the position-evaluation function (Section 4) in mind when formulating the position-generation procedure.

3. The candidate positions for a given label should occupy a variety of different locations near its feature in order to give the selection algorithm (Section 3) the opportunity to exploit tradeoffs between label positions for different features in its quest for a globally optimal labeling.

[3] This metric alone is completely adequate for the sample maps we consider in Section 6. However, it will clearly not suffice for all area-feature shapes and all position-generation algorithms. If necessary, additional metrics that are more sensitive to the shape of the area feature, e.g., metrics that encourage alignment with an area's medial axis (Ahn and Freeman, 1984), can be incorporated into the evaluation function. We omit such metrics from this discussion.

Overlap metrics (§4.1)	
PointOver	10
LineOver	15
AreaOver	10
LabelOver	40
Positioning metrics	
Point positioning (§4.2)	
PointPos	1
Line positioning (§4.3)	
AveDist	1
Flatness	1
MinDist	NA
Centeredness	3
Aboveness	0.25
Area positioning (§4.4)	
AreaPos	10

Table 1: Metric weights.

Note that it is not important that all the generated positions be of the highest quality, nor even that the best imaginable position be included. Instead, it is important that the set of positions includes a variety of locations, even if that requires inclusion of some mediocre positions. This is in contrast to other systems that first identify a single ideal position for a feature, and then choose minor perturbations of the position when attempting to resolve conflicts. Such strategies, by constraining the search space excessively, make the search for a globally optimal labeling much harder.

We describe position-generation procedures for the three fundamental types of map features: points, lines, and areas. These types are not entirely disjoint; for example, line features can sometimes be best described as area features, as with a river on a large-scale map. However, this is merely a problem of choosing the most appropriate fundamental type as a representation, not a new problem in position generation.

5.1 Candidate label positions for point features

Point-feature labels are generated according to the fixed pattern illustrated in Figure 2. Conceptually, this pattern is defined by matching anchor points on the label to anchor points on the point feature.

Anchor points of a point feature are at multiples of 45 degrees on a circle centered on the feature. The radius of this circle is given by $max(r * 1.3, r + 0.1 * f)$, where r is the radius of the point feature (or its effective radius if noncircular), and f is the width of 'x' in the label's font.

Anchor points of a label are on a box formed by the top and sides of the label's bounding box and the baseline. Horizontally, a point is located at an end, at the midpoint, or one-third of the way from an end. Vertically, a point is aligned with one of four lines: the top of the bounding box, a line through the tops of lower-case letters (the "x-height line"), a line centered between the

x-height line and the baseline, or the baseline. Not all combinations of anchor points are generated as candidate positions; the 17 positions used are shown in Figure 2.

In addition, label positions are altered with respect to per-letter kerning information kept in a lookup table. For instance, if a label is in the most favorable position, it will be slightly more to the left if the first letter is 'V' rather than 'D'. As noted in Section 4.2, candidate positions may also be altered to take descenders into account, thereby affecting the point-positioning metric values shown in the figure.

5.2 Candidate label positions for line features

The problem of generating positions for lines is thornier: unlike point features, line features appear in a rich variety of curves and orientations. We would prefer all candidate positions to avoid overlapping the features they tag. For point features, we can easily generate candidate positions that are guaranteed to be valid and of reasonable quality. With line features, however, we must be slightly more industrious in order to meet this proviso without paying an unreasonable price in performance.

Our approach is to apply a limited version of our overall generate/select paradigm before the general label-selection procedure is begun. Here is a summary:

1. Generate many candidate positions without worrying about their validity.
2. Adjust them in simple ways that optimize some of the more easily computed line metrics.
3. Evaluate them according to all precomputable line metrics.
4. Cull all but the k best positions for some k.

This procedure is clearly efficient, involves no time-consuming search, yet produces a variety of relatively good positions. Here is a more detailed version of the procedure.

1. Generate multiple positions along the length of the line feature:
   (a) Let *start* be the point at one end of the polyline.
   (b) Let *inc* be one eighth of *width*, the width of the label.
   (c) Repeat until *start* is less than *width* from the end of the polyline, measured as distance along the line:
      i. Find a point *end* on the polyline that is a distance *width* from *start*.
      ii. Generate two coincident positions with baselines that run from *start* to *end*; mark one "above", the other "below".
      iii. Increment *start* by *inc*.

   (At this point, the algorithm will have produced a large number of candidate positions, none of which will be positioned at a good distance from the line. In fact, most will intersect the line. However, the next step corrects this problem by applying a translation.)
2. Adjust all generated candidate positions to achieve the ideal value for the *MinDist* metric by applying appropriate translations perpendicular to their baselines. One position of each pair is moved to a position above the line; the other, below.

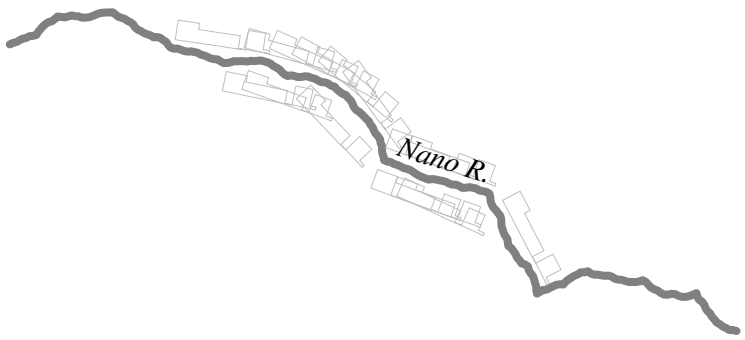

Figure 8: Generated label positions for a line feature.

3. If a line feature is shorter than its own label, no positions will have been generated in steps 1–2. If this is the case, pick a point at the center of the line feature and generate positions as if the given point were a point feature.

4. Score the generated positions according to all precomputable metrics — everything but *LabelOver*.

5. Delete all but the k best positions.

Typically, we use $k = 32$, which is a good compromise allowing a sufficient variety of positions while not overly expanding the search space. Figure 8 shows a typical line feature labeled at the best generated position, as well as the skylines of the 15 next-best positions.

5.3 Candidate label positions for area features

For many simple applications (car-navigation systems and the like), area features can be identified adequately by simple horizontal labels. We utilize the same strategy for areas that we did for lines: generate a large number of candidate positions, then cull by precomputable metrics.

Given a polygon P representing the area's boundary, we find the inset polygon P' such that if a label is centered at any point in P', the label will lie entirely inside P. (If no such P' exists, then it is impossible to place a label inside P that does not overlap the boundary of P. In this case, the area is labeled as if it were a point feature.) We then generate n (typically 200) quasirandom points evenly distributed throughout P' using a Sobol' quasirandom sequence (Press et al., 1992).[4] Candidate positions are generated centered at these 200 points. These positions are scored and all but the best-scoring k positions (again, we use $k = 32$) are eliminated.

Figure 9 shows the top 20 generated positions for the label of a given area feature; the actual label is shown at the most favorable position of the 20.

6 Sample Maps

As evidence that the labeling algorithm that we have presented has the potential to generate cartographically plausible labelings fully automatically and in reasonable time, we present a selection

[4]The n points are most useful if no two points are close together, because labels for nearby points are likely to overlap the same features and labels. The Sobol' quasirandom sequence is well suited for achieving a dispersed distribution of points.

Figure 9: Candidate label positions for an area feature.

of sample maps with randomly generated point, line, and area features as labeled by our prototype software system.[5] The sample maps exhibit the ability of the algorithm to trade off the evaluation criteria for point, line, and area features in a reasonable manner.

Figure 10 shows a randomly generated map with 300 point features, 10 line features, and one area feature. The map was labeled in eight seconds on a 200MHz Pentium PC. Only three of the eight seconds were spent doing position selection.[6]

Figures 11 through 13 show a typical progression of the position-selection process. Initially, labels are placed in random locations (Figure 11); in under half a second, most of the labels are positioned satisfactorily (Figure 12); and in under one second, the final labeling has been found (Figure 13).

[5]The point-feature coordinates were picked randomly from a uniform distribution. Points less than 0.1 inches from the center of any existing point feature were rejected so as to eliminate feature-feature overlaps. Additionally, a small number of points in configurations that precluded any valid labeling were removed or adjusted by hand. Line features were generated automatically by a fractal algorithm. Having a parameterized map-generation procedure that could produce maps with a variety of feature densities and characteristics was extremely useful in developing and testing our algorithm.

[6]Most of the time is spent on precomputation that facilitates efficient position evaluation. Several time-consuming aspects of this precomputation, primarily pairwise intersection testing, use inefficient algorithms that were selected primarily for their ease of implementation. A commercial-grade implementation on a state-of-the-art workstation should be able to label maps like the one in Figure 10 virtually instantaneously.

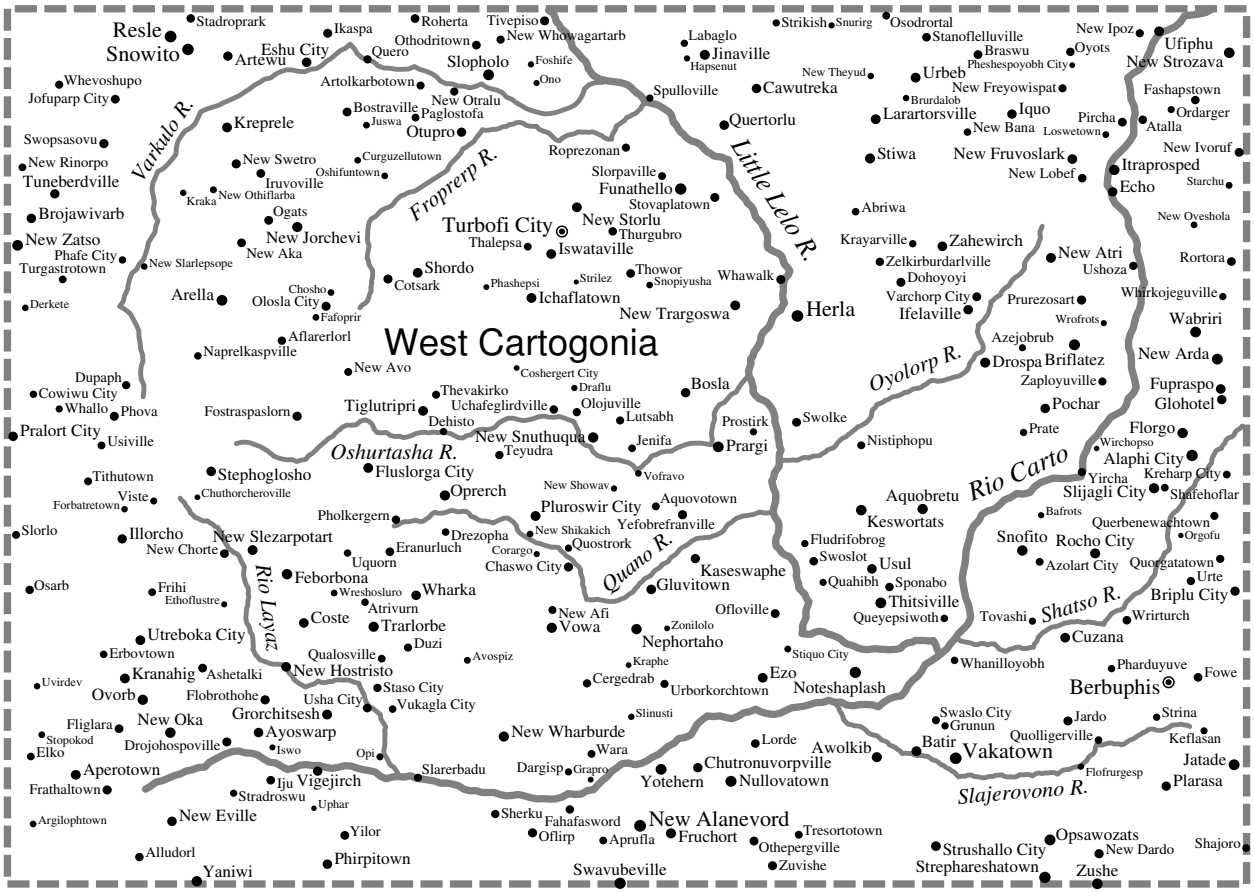

Figure 10: A randomly generated map labeled by the presented algorithm.

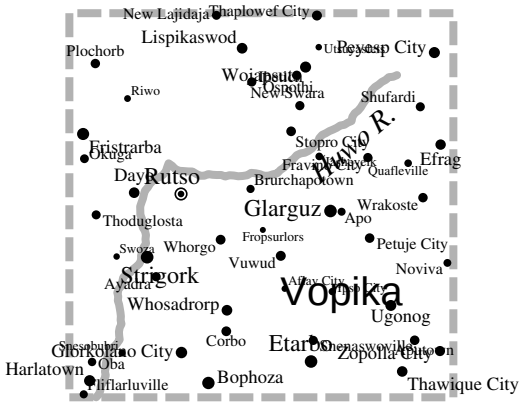

Figure 11: The initial random labeling of a randomly generated map.

Figure 12: The map of Figure 11, halfway through position selection.

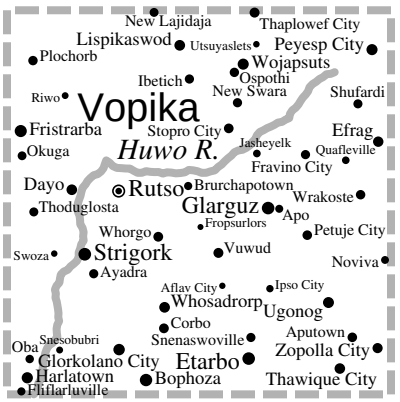

Figure 13: The final labeling of the map of Figure 11.

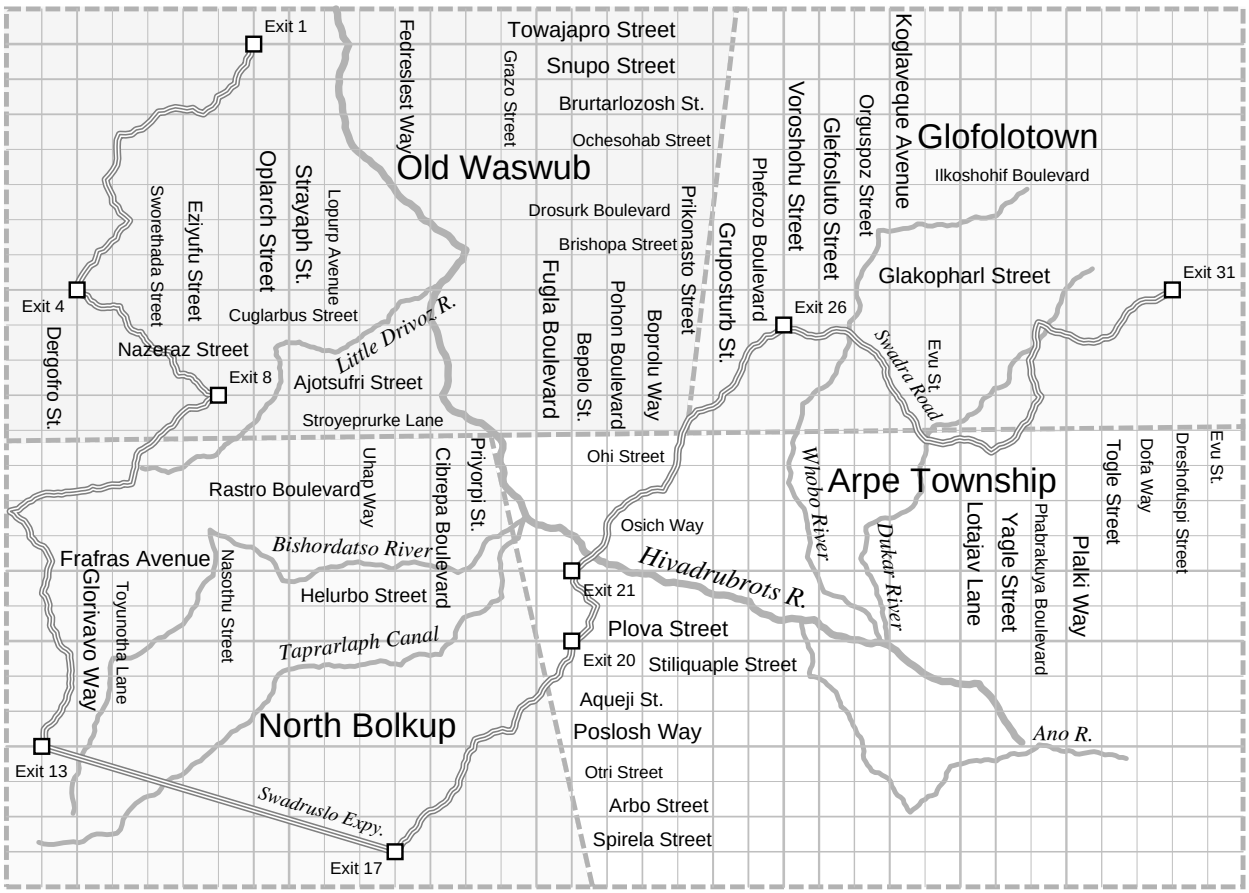

Figure 14: A random map with line and area features.

A different type of map is shown in Figure 14. In contrast to the map in Figure 10, this map contains mostly line and area features, and shows the ability of the algorithm to find a plausible labeling under the tight constraints presented by the interaction of these features.

7 Conclusions and Future Work

We have presented a cartographic labeling algorithm that is:

- *general* — point, line, and area features are labeled using a unified methodology;
- *effective* — it produces compact, visually appealing maps;
- *efficient* — text placement is accomplished in less than 10 seconds on a standard personal computer for page- or screen-sized maps with up to 300 labeled features;
- *accessible* — it is simply explained, easily implemented, and readily extended.

Our implementation of the algorithm illustrates the advantages to be gained from a framework in which cartographic knowledge is summarized numerically in a scoring function that is then optimized by powerful heuristic search techniques.

While we believe this framework to be far superior to previous approaches that are based on expert-system production rules and depth-first search, we do not claim that the specific algorithm described here cannot be improved. For example, the following problems warrant further attention:

- *Disambiguation:* Our current scoring function classifies the interaction between each label and feature as either overlapping or nonoverlapping. However, more subtlety is necessary to identify candidate positions that do not overlap features, but that come close enough to cause some degree of ambiguous association.
- *Selectivity:* A useful option in labeling dense maps is to allow for (occasional) deletion of certain labels or even the features themselves in areas of congestion. As noted above, we had to resort to some manual feature deletion in our sample maps to ensure that an acceptable labeling was feasible. Automation of this selection process has already been demonstrated in the context of point-feature label placement (Langran and Poiker, 1986; Christensen, Marks, and Shieber, 1995).
- *Flexibility:* When label or feature deletion is not acceptable, the text can be modified to make the labeling problem simpler, or at least to provide additional placement options. For example, labels can be replaced by numbers (with a suitable key shown elsewhere), and text can be broken and stacked. Another option is to connect displaced labels and the features they tag by a line.
- *Expressivity:* Candidate position suggestion should allow for greater expressivity. For instance, horizontal labels are not ideal for all area features, curved labels may be better for some line and area features, and long linear features should allow for multiple occurrences of the label and for distribution of multi-word labels along the length of the feature.

These problems present no conceptual difficulty for the algorithm outlined here. We anticipate that they can be addressed by improved generation procedures and evaluation metrics without a significant increase in execution time. In the hope that others might want to build on our work, we are happy to make our software available on request to other researchers. Our testbed is written in C, runs on Unix platforms, and produces PostScript output.

8 Acknowledgments

Much of the work described in this paper was done while Christensen and Edmondson were at Harvard University and Marks was at Digital Equipment Corporation's Cambridge Research Lab. This research was supported in part by Grant Number IRI-9350192 from the National Science Foundation, and by grants from Digital Equipment Corporation, Xerox Corporation, and MERL – A Mitsubishi Electric Research Laboratory.

References

Ahn, John and Herbert Freeman. 1984. A program for automatic name placement. *Cartographica*, 21(2–3):101–109, Summer–Autumn.

Christensen, Jon, Joe Marks, and Stuart Shieber. 1993. Algorithms for cartographic label placement. In Gary G. Kelly, editor, *Proceedings of the American Congress on Surveying and Mapping '93, Vol. 1*, pages 75–89, New Orleans, Louisiana, February.

Christensen, Jon, Joe Marks, and Stuart Shieber. 1995. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14(3):203–232, July.

Doerschler, Jeffrey S. and Herbert Freeman. 1992. A rule-based system for dense-map name placement. *Communications of the Association for Computing Machinery*, 35(1):68–79, January.

Ebinger, Lee R. and Ann M. Goulette. 1990. Noninteractive automated names placement for the 1990 decennial census. *Cartography and Geographic Information Systems*, 17(1):69–78, January.

Feigenbaum, Mitchell. 1994. Method and apparatus for automatically generating symbol images against a background image without collision utilizing distance-dependent attractive and repulsive forces in a computer simulation. U.S. Patent #5,355,314, filed 11/5/93, and granted 10/11/94. Assigned to Hammond Inc., Maplewood, New Jersey, October.

Hirsch, Stephen A. 1982. An algorithm for automatic name placement around point data. *The American Cartographer*, 9(1):5–17.

Imhof, Eduard. 1962. Die Anordnung der Namen in der Karte. *International Yearbook of Cartography*, 2:93–129.

Jones, Christopher B. 1989. Cartographic name placement with Prolog. *IEEE Computer Graphics and Applications*, 9(5):36–47, September.

Kirkpatrick, S., C. D. Gelatt Jr., and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220:671–680.

Langran, Gail E. and Thomas K. Poiker. 1986. Integration of name selection and name placement. In *Proceedings of the Second International Symposium on Spatial Data Handling*, pages 50–64, Seattle, Washington, July. International Geographical Union and International Cartographic Association.

Press, William, Saul Teukolsky, William Vetterling, and Brian Flannery. 1992. *Numerical Recipes in C*. Cambridge University Press, Cambridge, England, 2nd edition.

Zoraster, Steven. 1986. Integer programming applied to the map label placement problem. *Cartographica*, 23(3):16–27.

Zoraster, Steven. 1990. The solution of large 0-1 integer programming problems encountered in automated cartography. *Operations Research*, 38(5):752–759, September-October.

Zoraster, Steven. 1991. Expert systems and the map label placement problem. *Cartographica*, 28(1):1–9, Spring.

Contents

1 Introduction **1**

2 Technical Approach **2**

3 Position Selection **3**

4 Position Evaluation **4**
4.1 Overlap metrics 4
4.2 Point-positioning metrics 5
4.3 Line-positioning metrics 6
4.4 Area-positioning metrics 9
4.5 The overall evaluation function 9

5 Position Generation **9**
5.1 Candidate label positions for point features 10
5.2 Candidate label positions for line features 11
5.3 Candidate label positions for area features 12

6 Sample Maps **12**

7 Conclusions and Future Work **16**

8 Acknowledgments **17**