Labeling Point Features on Maps and Diagrams

Labeling Point Features on Maps and Diagrams

Jon Christensen
Harvard University
Cambridge, Massachusetts

Joe Marks
Mitsubishi Electric Research Laboratories
Cambridge, Massachusetts

Stuart Shieber
Harvard University
Cambridge, Massachusetts

Abstract

A major factor affecting the clarity of graphical displays that include text labels is the degree to which labels obscure display features (including other labels) as a result of spatial overlap. Point-feature label placement (PFLP) is the problem of placing text labels adjacent to point features on a map or diagram so as to maximize legibility. This problem occurs frequently in the production of many types of informational graphics, though it arises most often in automated cartography. In this paper we present a comprehensive treatment of the PFLP problem, viewed as a type of combinatorial optimization problem. Complexity analysis reveals that the basic PFLP problem and most interesting variants of it are NP-hard. These negative results help inform a survey of previously reported algorithms for PFLP; not surprisingly, all such algorithms either have exponential time complexity or are incomplete. To solve the PFLP problem in practice, then, we must rely on good heuristic methods. We propose two new methods, one based on a discrete form of gradient descent, the other on simulated annealing, and report on a series of empirical tests comparing these and the other known algorithms for the problem. Based on this study, the first to be conducted, we identify the best approaches as a function of available computation time.

CR Categories: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*geometric problems and computations, routing and layout*. H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*screen design*. 2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*cartography.* I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*geometric algorithms, languages, and systems*.

General Terms: algorithms, experimentation.

Additional Key Words and Phrases: label placement, automated cartography, stochastic methods, simulated annealing.

1 Introduction

Tagging graphical objects with text labels is a fundamental task in the design of many types of informational graphics. This problem is seen in its most essential form in the field of cartography, where text labels must be placed on maps while avoiding overlaps with cartographic symbols and other labels, though it also arises frequently in the production of other graphics (e.g., scatterplots). Although several techniques have been reported for automating various label-placement tasks, the positioning of labels is still performed manually in many applications, even though it can be very tedious. (Cook and Jones (1990) report that cartographers typically place labels at the rate of only 20 to 30 labels per hour, with map lettering contributing up to half of the time required for producing high-quality maps.) Determining an optimal positioning of the labels is, consequently, an important problem.

In cartography, three different label-placement tasks are usually identified: labeling of area features (such as oceans or countries), line features (such as rivers or roads), and point features (such as cities or mountain peaks) (Imhof, 1962; 1975). While it is true that determining the optimal placement of a label for an isolated point feature is a very different task from determining the optimal placement of a label for an isolated line or area feature, the three placement tasks share a common combinatorial aspect when multiple features are present. The complexity arises because the placement of a label can have global consequences due to label-label overlaps. This combinatorial aspect of the label-placement task is independent of the nature of the features being labeled, and is the fundamental source of difficulty in automating label placement. We therefore

concentrate on point-feature label placement (PFLP) without loss of generality; in Section 5 of the paper we describe how our results generalize to labeling tasks involving line and area features.

The PFLP problem can be thought of as a combinatorial optimization problem. Like all such problems, two aspects must be defined: a *search space* and an *objective function.*

Search space. An element of the search space can be thought of as a function from point features to label positions, which we will call a *labeling*. The set of potential label positions for each point feature therefore characterizes the PFLP search space. For most of the published algorithms, the possible label positions are taken, following cartographic standards, to be a finite set, which is enumerated explicitly. Figure 1 shows a typical set of eight possible label positions for a point feature. Each box corresponds to a region in which the label may be placed. Alternatively, a continuous placement model may be used, for example by specifying a circle around the point feature that the label must touch without intersecting.

In certain variants of the PFLP problem, we allow a labeling not to include labels for certain points (presumably those that are most problematic to label, or least significant to the labeling application). When this option is included, the PFLP problem is said to include *point selection*.

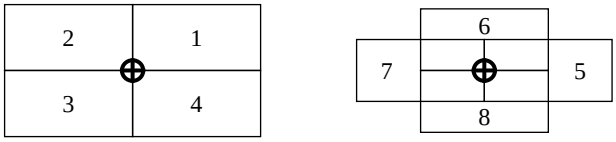


Figure 1: A set of potential label positions and their relative desirability. Lower values indicate more desirable positions.

Objective function. The function to be optimized, the objective function, should assign to each element of the search space (a potential labeling of the points) a value that corresponds to the relative quality of that labeling. The notion of labeling quality has been studied by cartographers, most notably by Imhof (1962; 1975). However, Imhof's analysis is descriptive, not prescriptive; coming up with an appropriate definition of the objective function for a general label-placement problem (that is, one that includes point, line, and area features) is a difficult task. Labeling quality can depend on many factors, including detailed "world knowledge" and characteristics of human visual perception. Many of the label-placement algorithms reported in the literature therefore incorporate sophisticated objective functions. A popular approach has been to use a rule-based paradigm to encode the knowledge needed for the objective function (Ahn and Freeman, 1984; Freeman and Ahn, 1987; Jones, 1989; Cook and Jones, 1990; Doerschler and Freeman, 1992). For the PFLP problem, however, a relatively simple objective function suffices. Our formulation of the objective function is due to Yoeli (1972)[1]. In Yoeli's scheme, the quality of a labeling depends on the following factors:

- The amount of overlap between text labels and graphical features (including other text labels);
- A priori preferences among a canonical set of potential label positions (a standard ranking is shown in Figure 1); and
- The number of point features left unlabeled. (This criterion is pertinent only when point selection is incorporated into the PFLP problem.)

Figure 2 provides an illustration of these factors. By specifying how to compute a numerical score for each of the criteria above, an objective function can be defined. Such a function assigns to each labeling a number that indicates its relative quality. We will assume that low scores correspond to better labelings, so that the goal of the search is to minimize the objective function.

The PFLP problem is a combinatorial optimization problem defined by its search space and objective function; a solution to the problem is comprised of a search algorithm that attempts to find a relatively good element of the search space. A natural issue to raise, before exploring possible search algorithms, is the intrinsic complexity of this search problem. In Section 2 we

[1]A recent study conducted by Wu and Buttenfield (1991) addresses the issue of placement preference for point-feature labels in more detail.

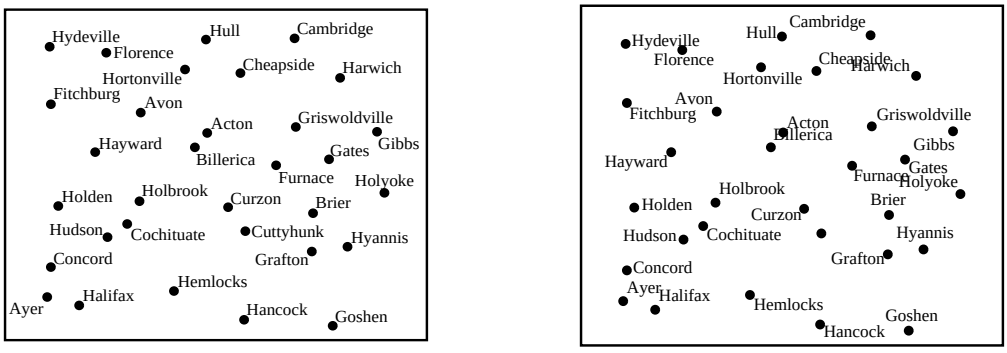


Figure 2: Good (a) and bad (b) labelings of the same map.

summarize some previous results that show that the problem and many of its interesting variants are NP-hard. Thus, any complete search algorithm will be intractable, any tractable algorithm incomplete.[2]

This characterization is borne out by previously published algorithms, which fall into two classes: exhaustive search algorithms and local search algorithms. We review these algorithms in Section 3. As expected, the exhaustive algorithms are computationally profligate, and the local search algorithms are incomplete, in that they tend to find local, rather than global minima.

We also present two new algorithms for the PFLP problem in Section 3. The first is a local search technique based on a discrete form of gradient descent. Although it is also incomplete, its performance on problems with high label density and its efficiency make it attractive under certain circumstances. The second technique is a stochastic algorithm based on simulated annealing. An extensive empirical comparison of all the algorithms, the first comparative study of label-placement heuristics, is presented in Section 4;[3] it illustrates the advantages of the new methods and provides recommendations for selecting a labeling algorithm.

2 The Computational Complexity of PFLP

In this section, we review some recent results on the inherent complexity of PFLP that have implications for algorithm design. To demonstrate the inherent complexity of the problem (and, subsequently, to compare various algorithms for the task), we must decide upon a particular instance of search space and objective function. We begin with a relatively simple version of the problem. Once this simplified problem is shown to be NP-hard, it is straightforward to demonstrate that more complicated variants of the problem are also NP-hard. Our initial statement of the PFLP problem relies on the following simplifying assumptions:

Search space: discrete placement model comprising four equally favored candidate positions — those numbered 1 through 4 in Figure 1 — is used.

Objective function: The objective function to be minimized is the number of point features labeled with one or more overplots. Point selection is not allowed.

This simplified PFLP problem is an optimization problem. In order to apply the theory of NP-completeness to PFLP (Garey and Johnson, 1979), we formulate a corresponding decision problem. For any given PFLP problem instance, we can ask the question: Is there an admissible labeling, a labeling with a score of zero, in which no labels overlap and no point features are obscured? The NP-completeness of this admissible-labeling problem has been established independently by at least three different teams of researchers (Kato and Imai, 1988; Marks and Shieber, 1991; Formann and Wagner, 1991).

[2]This holds, of course, only if P ≠ NP, as is commonly believed.

[3]Brief summaries of this work have appeared elsewhere (Christensen 1993; 1994).

An algorithm for the PFLP optimization problem could always be used to solve the admissible-labeling problem: find an optimal labeling and check to see whether the cost is 0. Thus the PFLP optimization problem is at least as difficult as the admissible-labeling problem; in other words, the admissible-labeling result implies that optimal PFLP is NP-hard.

If label sizes are held steady, increasing the scale of a map makes more room for labels. This observation leads to the following question: How much must the scale be increased to permit an admissible labeling for a given PFLP problem instance? Formann and Wagner have developed an efficient algorithm for this problem that is guaranteed to find an admissible labeling with a map scale no more than twice optimal (Formann and Wagner, 1991). In spite of the apparent intractability of the basic problem, some simple restrictions can reduce the complexity dramatically. For example, a placement model that allows only two potential positions for each label results in a problem that is solved easily in polynomial time (Formann and Wagner, 1991). Similarly, the restricted set of problem instances in which no potential label position overlaps more than one other potential label position can also be solved efficiently.[4] Unfortunately, these polynomially solvable subcases are too simple to be of much practical interest.

The recent complexity results make it clear that PFLP is almost certainly intractable. Thus the failure of previous researchers to find an exact, tractable algorithm for PFLP is not surprising — it is extremely unlikely that anyone will ever discover such an algorithm. Instead, research efforts should be directed towards powerful heuristic methods that do not have guaranteed performance bounds, but that may work acceptably in practice. Several such algorithms are described and compared in the next section.

3 Algorithms for PFLP

Previously proposed PFLP algorithms fall into two main classes: those that perform a potentially exhaustive global search for an acceptable or optimal labeling, and those that perform search on a local basis only.

3.1 Exhaustive search versus local search

Exhaustive search algorithms for constraint satisfaction are often categorized as either brute-force or heuristic, depending on the manner in which backtracking is performed (Korf, 1988). As an example of brute-force backtracking, consider an algorithm that enumerates points in a prescribed order and places each label in a position which is currently unobstructed. If, as the algorithm proceeds, a point cannot be labeled (either because there are no positions without conflict, or because all available positions have been tried), the algorithm returns to the most recently labeled point and considers the next available position. The algorithm continues in this way until an acceptable labeling is identified or until the entire search space has been exhausted. A variety of modifications can be made to this algorithm in the hope of improving its performance. Standard techniques for reducing backtracking include variable ordering (Freuder, 1982; Purdom, 1983), value ordering (Dechter and Pearl, 1985; Haralick and Elliot, 1980), and returning to the source of failure (Gaschnig, 1979).[5] Another category of heuristics that may be considered for exhaustive search algorithms are those that attempt to prune the search tree in order to manage the complexity. These techniques, as they apply to label placement, are described below.

Variable ordering

One general method of minimizing backtracking is to make more constrained choices first; this is achieved by earlier instantiation of the more tightly constrained variables in the search problem. The application of this idea in the label-placement domain is straightforward: instead of considering features in an arbitrary order, a subsequent point might be chosen based on the number of labeling options available for it, or on the number of labeling options for other features that its label could eliminate. By labeling the most difficult features first, large portions of the search space may be ruled out before many choices have been made, thus reducing backtracking.

Value ordering

A second method of intelligent backtracking affects the order in which values are chosen for variables in the search problem. The application of this idea to label placement suggests a prioritization of potential label positions for a given feature in an attempt to identify innocuous label placements and avoid later conflicts. This method minimizes backtracking by attempting

[4] Developing an efficient algorithm for this artificial problem is left as an exercise for the interested reader.

[5] Korf gives a more comprehensive survey of general techniques (Korf, 1988).

labelings that are most likely to succeed first, in an effort to find a solution earlier in the search process. One common method is to choose between potential label positions on the basis of which position has the fewest number of potential conflicts with other features and labels.

Source of failure

In brute-force backtracking, when an impasse is reached the algorithm backtracks to the most recently placed label and chooses the next available position. This label often has little to do with the inability to label the current point, however. Instead, backtracking may be directed to one of the features conflicting with the current feature. If point selection is allowed, a similar insight applies: When backtracking has exhausted the search space, instead of deleting the last successfully labeled node, an effort is made to identify the most troublesome feature causing conflict with the unlabelable point. Although reported exhaustive-search algorithms for PFLP do not incorporate source-of-failure techniques for backtracking, we mention the technique for completeness.

Pruning heuristics

Pruning heuristics do not affect the exponential nature of the search space, but seek to make it as tractable as possible by eliminating areas of the search space from consideration. A common example of this is the elimination of label positions which exceed a predefined density threshold. If choosing a particular label position would eliminate more than a fixed number of surrounding label positions, then the offending position is eliminated from consideration. If all potential label positions of a feature are eliminated in this way, the feature is deleted. In case an acceptable labeling is not identified under this pruning regime, the process can be repeated with a lower threshold until a solution can be found. A situation for which this pruning heuristic is well suited is when a single particularly offensive label position conflicts with a large number of surrounding positions that are otherwise mutually disjoint.

Summary

Guided depth-first search with backtracking — the general paradigm just described — has formed the basis for numerous reported algorithms for label placement (Ahn and Freeman, 1984; Mower, 1986; Freeman and Ahn, 1987; Noma, 1987; Freeman, 1988; Jones, 1989; Cook and Jones, 1990; Ebinger and Goulette, 1990; Doerschler and Freeman, 1992; Consorti et. al, 1993). While these algorithms perform acceptably for relatively small problems, in practice the exponential nature of the search space quickly overcomes the heuristics for even moderately sized problems, making the approach of exhaustive search impractical as a general solution to the PFLP problem. Although the complexity results demonstrate that any exhaustive method is inevitably intractable, we include these algorithms in the summary because tractable, nonexhaustive variants can be constructed from them. (See Section 3.2.) The widespread use of exhaustive search techniques for the combinatorial aspects of the label-placement problem is indeed something of a mystery, given the availability of much better algorithms. Zoraster (1991) notes that part of the problem might be the inappropriate use of expert-system technology: whereas a rule-based approach is useful in general label placement for computing potential label positions and for evaluating candidate labelings, it suggests, misleadingly, that rule-based techniques — exhaustive search is easy to implement in a rule-based system — are useful for all aspects of label placement.

3.2 Greedy algorithms

A more practical approach to search results from avoiding the unbounded backtracking strategy of the exhaustive methods altogether. By limiting the scope of the search, more efficient algorithms can be devised. Of course, these algorithms may not find optimal solutions, but the hope is that a suitable trade-off between labeling quality and computational cost can be found.

Instead of undoing previously computed label placements, as in guided depth-first search with backtracking, any point whose label cannot be placed can be treated summarily: the point can be left out if point selection is allowed (Langran and Poiker, 1986), or it can be labeled even though a label overlap or feature obscuration results. (A third option, that of appealing to a human oracle for assistance, is noted by Yoeli (1972) as a practical alternative.) Such a “greedy algorithm” for PFLP yields behavior that is tractable for a much more realistic space of problems, although the lack of backtracking certainly impairs the quality of the solutions that are found. For a greedy algorithm to be at all effective in identifying reasonable labelings, it is essential that heuristics for guiding the search, such as those described in Section 3.1, be used. Even then, there is typically much improvement that can be made to the resulting labelings, as will be shown subsequently.

3.3 Discrete gradient descent

The quality of labelings produced by a greedy algorithm can be improved dramatically if the labelings are repaired subsequently by local alteration. This is the motivation for the gradient-descent algorithms presented below. A gradient-descent method is defined relative to a set of operations that determine how an existing labeling can be altered by specifying ways in which one or more labels can be repositioned simultaneously. The basic idea of gradient descent is to choose from among the set of available operations the one that yields the most immediate improvement. By repeatedly applying the operation that most improves the labeling (or, equivalently, the operation that causes the most movement in the direction of the objective-function gradient), a new labeling can be computed that is significantly superior to the original. Again we present a straw man to exemplify the idea. Let the set of operations comprise those that move a single label arbitrarily from one potential position to another. An outline of the resulting algorithm, which we call *discrete gradient descent* is given below:

1. For each feature, place its label randomly in any of the available candidate positions.
2. Repeat until no further improvement is possible:
   (a) For each feature, consider moving the label to each of the alternative positions.
   (b) For each such repositioning, calculate the change in the objective function which would result if the label were moved.
   (c) Implement the single label repositioning that results in the most improvement. (Ties are resolved randomly.)

In practice the algorithm precomputes a table of costs associated with each possible repositioning. After each label positioning, only elements of the table that touch the old or new label positions are recomputed.

Local minima

Clearly, the major weakness of the discrete gradient-descent algorithm is its inability to escape from local minima of the objective function. Figure 3 shows a typical example of a local minimum. In this case, the conflict can be resolved by moving the lower feature's label to its bottom-left position and the upper feature's label to its upper-right position. Unfortunately, making any single move has no effect on the value of the objective function, and, because the algorithm only accepts changes which show an immediate improvement, the algorithm is unaware of the possibility of accepting a neutral move in order to make an improvement. Adjusting the algorithm to allow it to make moves that do not affect the objective function might remedy this particular example, but is not sufficient in general. In the example of Figure 4, the current value of the objective function could be improved from 4 (Figure 4a) to 3 (Figure 4b) by moving the four middle labels to their left-most positions. However any one of these moves will initially result in an uphill step and an intermediate score of 5. To limit the incidence of such local minima, more sophisticated gradient descent heuristics have been devised. Nevertheless, as we will see, the discrete-gradient descent method performs surprisingly well given its naivete.

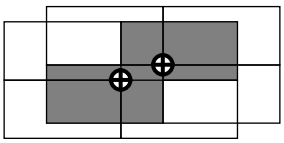


Figure 3: A local minimum of the discrete gradient-descent algorithm. The candidate label positions are marked with boxes, and the selected label positions are shaded.

3.4 Approximating the gradient with overlap vectors

Hirsch (1982) presents a more sophisticated gradient descent method for PFLP. Hirsch's algorithm uses a continuous placement model in which each point feature has an infinite set of potential label positions. The potential positions for a point touch, but do not intersect, a circle centered about the point; labels are allowed to slide continuously around a circle (see Figure 5a). When the label touches at the highest, lowest, left-most, or right-most points of the circle, it is considered to be in a special zone and is allowed to slide back and forth along the point of tangency (see Figure 5b).

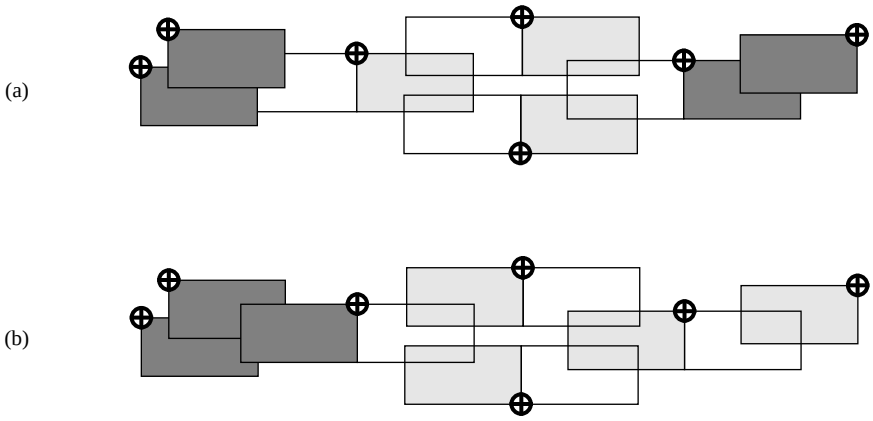


Figure 4: Another local minimum of the discrete gradient-descent algorithm (a) and an optimal configuration (b). The candidate label positions are marked with boxes, and selected label positions are shaded. Obstructed label positions are shaded dark.

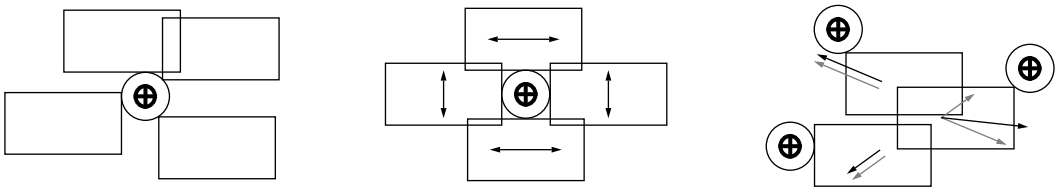


Figure 5: Some example potential label positions for Hirsch's algorithm (a), along with the special zones (b), and an example of overlap vectors (c)

Initially each label is placed in the special zone to the right of its point. Each label is then tested for overlaps with other labels and intersections with the circular boundaries of other points. For each conflict an overlap vector is computed based on the *x* and *y* extents of the overlap or intersected area. Each overlap vector is split between the two conflicting features and represents the movement required to eliminate a particular conflict. The sum of overlap vectors associated with each label is then calculated to give an aggregate vector that represents (in an intuitive sense) a good direction in which to move the label so as to eliminate the overlaps and intersections. In Figure 5c the overlap vectors are drawn in light gray, and the aggregate vectors in black. (For labels involved in only one conflict the single overlap vector and the aggregate vector are the same.)

Once an aggregate overlap vector has been calculated for each label, the algorithm seeks to move each label in the general direction of this vector in an effort to generate a labeling with fewer overlaps. The heuristic technique employed involves two styles of movement, a gradual movement around the circle and a more abrupt movement which shifts the label directly to the point on the circle indicated by the overlap vector. Thus there are only two operations available for altering a labeling, but each operation is applied to all point features on a given round of application so that many labels may change positions simultaneously. The gradual-style movement involves a series of heuristic rules that specify whether the label should be repositioned exactly as the aggregate vector indicates, whether only one component of the aggregate vector should be used to reposition the label (e.g., if it is in a special zone), or whether the label should simply be moved to the nearest special zone in the direction of the aggregate vector (e.g., if the vector indicates the label should be positioned outside of the current quadrant). Hirsch suggests alternating between the two movement styles, with more frequent application of the gradual-style movement.

The intuition behind the algorithm is best explained by an analogy with a physical system. The individual overlap vectors represent a "force" of repulsion between overlapping objects, the sum an aggregate force. Thus, through gradual movements, the system settles into a local minimum of the "energy" of the system. The overlap vectors approximate the gradient in the energy space. To allow some ability to exit from local minima, the abrupt movements are designed to allow a jump from one energy state to another, hopefully lower one.

There are two sources of problems for Hirsch's algorithm. First, since the overlap vectors provide only an approximation of the gradient, they are subject to error. Second, like the discrete gradient-descent algorithm, Hirsch's algorithm is susceptible to getting stuck in local minima.

Gradient approximation errors

A typical dilemma is due to the summation of overlap vectors. When multiple labels overplot a single label, the magnitude of the calculated aggregate vector will often be unnecessarily large, leading to problems of overshooting during gradual-style movements.

Note also that Hirsch's overlap vectors each exhibit two degrees of freedom, whereas the labels are constrained to lie tangent to their associated circles. The result is that even in those cases where the accumulated overlap vector represents a favorable direction of movement, the particular manner in which a label is repositioned is often quite fragile. If a large component of the overlap vector points radially outward, for example, the location of the repositioned label is somewhat arbitrary.

Local minima

Hirsch's algorithm, like the discrete gradient-descent algorithm, can also get stuck in local minima. The nature of these minima is closely related to the specific heuristics the algorithm employs in response to various overlap situations. Figure 6 shows a problematic configuration. During applications of gradual-style movement, the label is adjusted slightly up and down until it is centered between, but still conflicting with, the two labels above and below. During applications of the abrupt-style movement, the horizontal component of the overlap vector dominates, and the label cycles between the left and right placements, missing the acceptable positions above and below the feature.

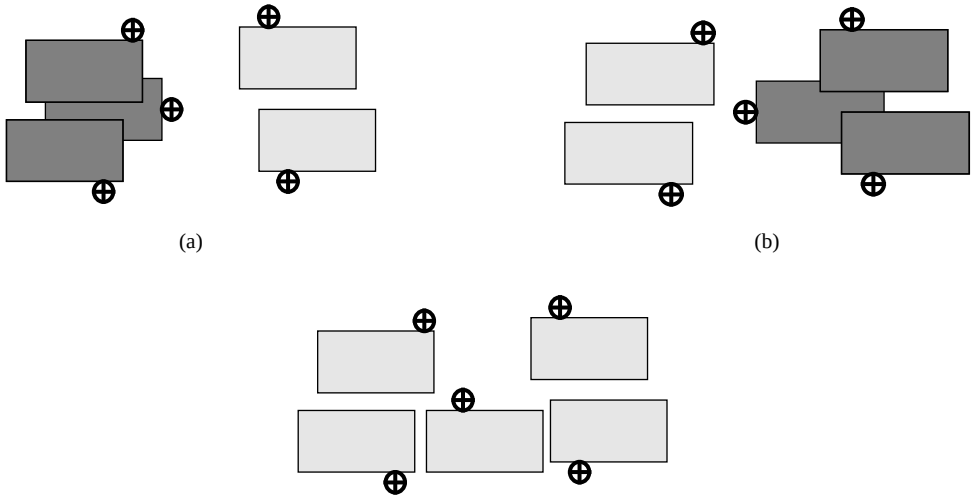


Figure 6: A local minimum of Hirsch's algorithm. The algorithm oscillates between configurations (a) and (b), unable to discover the preferred configuration (c).

3.5 Implementation details for Hirsch's algorithm

Calculation of overlap vectors

In the implementation of Hirsch's algorithm, we noted a subtle point in the vector computation not described in his paper. The overlap vectors must be calculated with reference to both the relative positions of the features and the conflicting label positions; in many situations where features are closely placed, an overlap calculation based solely on the overlap geometry results in overlap vectors which push the labels in unattainable directions. Figure 7 shows an example of this: both conflicts exhibit identical overlaps, yet because of the relative positions of the features, distinct overlap vectors are required. Similar situations arise in the calculation of overlap vectors between labels and the placement circles of point features. Without this addition to the overlap-vector calculation, labels are often trapped by neighboring points; once this occurs the overlap vector and label position are in the same quadrant and, given Hirsch's heuristics, the conflict will remain for the duration of the algorithm.

(a) (b)

Figure 7: Computation of overlap vectors in Hirsch's algorithm. The configurations in (a) and (b) have identical overlap regions, but the overlap vectors must be calculated to cause movement in different directions due to the relative placement of the overlap and the points.

Compensating for the placement model

In order to compare the performance of Hirsch's algorithm against other PFLP algorithms, several issues relating to the placement model need to be addressed. The presence of a circular buffer surrounding each point feature handicaps the algorithm, disallowing free space that other algorithms are able to exploit and forcing labels outward, increasing their effective dimensions. We considered two methods to compensate for this. First, we experimented with adjusting the label sizes for Hirsch's algorithm. We decreased the dimensions of each label such that the combined area of the placement circle and reduced label was equivalent to the area of the unmodified label. Second, we simply set the radius of the placement circle to zero. We found the latter method to perform slightly better on average, and included this variant of the algorithm in our comparisons.[6] This had the additional advantage of reducing angular computations and improving the run-time performance of the algorithm.

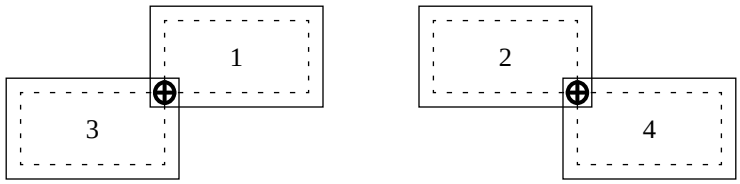


Figure 8: Unambiguous label regions for a point. By expanding the region reserved for a label to include regions of the plane beyond those covered by the label itself, labels will be unambiguously associated with a point feature at the expense of a higher effective density and therefore a more difficult labeling problem

The presence of the placement circle should not be completely discounted, however, as a primary reason for its inclusion is to provide unambiguous label-point association. If overplots between a placement circle and other labels are eliminated, then the designated feature label, which lies on the circle, will be the closest label to its feature. Figure 8 shows one way in which this effect can be approximated using a discrete placement model by expanding the region reserved for a label.
A related issue involves the continuous nature of the placement model. Since this allows a larger and therefore less-constrained search space, this probably gives Hirsch's algorithm an advantage. Although this discrepancy is harder to resolve, a fairer comparison could be obtained by running the discrete algorithms with a 16 or 20-position placement model, as opposed to the four-position model used in the experiments. However, the results described in Section 4 render this point irrelevant.

3.6 Mathematical programming for PFLP

Next, we turn to an algorithm introduced by Zoraster (1986; 1990) that addresses the optimization nature of PFLP directly by applying mathematical programming techniques to its solution.[7] Zoraster begins by formulating PFLP as a 0-1 integer programming (ZOLP) problem:[8]

[6]This is perhaps not surprising given the algorithm's predilection for label placements within special zones. Gradual-style movements tend to relocate labels into special zones, whereas only application of the abrupt movement style is able to move a label out of a special zone. Since the algorithm finishes with a series of 15 gradual-style movements, in practice nearly all labels finish in special zones.

[7]This algorithm is in commercial use in the oil industry to label drilling maps (Zoraster, 1990).

[8]Cromley (1986) has experimented independently with a slightly different ZOLP formulation of the label-placement problem.

- Given K labels and N_k possible positions for each label, each potential label position is represented by a variable $X_{i,k}$, $1 \le i \le N_k$, and $0 \le k \le K$. (Point selection is achieved by specifying a special label "position" that indicates a deselected point.)
- Each $X_{i,k}$ has value 0 or 1, indicating the absence or presence, respectively, of a label in that position.
- One set of constraints expresses the requirement that each point be labeled exactly once: $\sum_{i=1}^{N_k} X_{i,k} = 1$ for $1 \le k \le K$.
- Given Q pairwise overlaps between possible label positions, a second set of constraints expresses the requirement that no two labels overlap: $X_{r_q, s_q} + X_{r'_q, s'_q} \le 1$ for each potential overlap, $1 \le q \le Q$.
- The objective function is $\sum_{k=1}^{K} \sum_{i=1}^{N_k} W_{i,k} \times X_{i,k}$, where $W_{i,k}$ is a weighting that represents placement preferences.

Because ZOLP is itself NP-hard (Karp, 1972; Sahni, 1974), a complete, efficient algorithm for the PFLP problem recast in this way is still not possible, but heuristic techniques for ZOLP can now be applied to the PFLP problem. Zoraster combines Lagrangian relaxation, subgradient optimization, and several problem-specific heuristics in his solution. The primary insight of Zoraster's algorithm is to relax the overplot constraints and include them as additional penalty terms in the objective function. This gives:

- Minimize $\sum_{k=1}^{K} \sum_{i=1}^{N_k} W_{i,k} \times X_{i,k} + \sum_{q=1}^{Q} (X_{r_q, s_q} + X_{r'_q, s'_q} - 1)\, d_q$
- Still subject to $\sum_{i=1}^{N_k} X_{i,k} = 1$ for $1 \le k \le K$

In this modified objective function, the $d_q \ge 0$ are Lagrangian multipliers, one for each pairwise overplot constraint. Note that for a given set of Lagrangian multipliers, the minimum value of the objective function is easily identified by choosing the label-position variable with the smallest objective-function coefficient for each point feature. Although Lagrangian methods for ZOLP can be arbitrarily sophisticated, Zoraster's basic algorithm is a straightforward implementation of standard techniques (Fisher, 1981):

1. Compute and store the objective-function coefficient for each potential label position.
2. Generate a current labeling (CL) by picking the label position with the lowest objective-function coefficient for each point feature.
3. Initialize the active constraint set (ACS) to the empty set.
4. Repeat for 40 iterations or until a solution with no label conflicts is found:
   (a) Identify all pairwise constraints that CL violates and add any new ones to ACS. (The Lagrangian multiplier of each newly introduced constraint is zero initially, so adding a new constraint to ACS does not affect the objective-function coefficients.)
   (b) Make a local copy, CL', of CL.
   (c) Repeat for *x* iterations, where *x* is the lower of 400 or the number of iterations required to find a feasible solution with respect to the current ACS, plus an additional 100 iterations if a feasible solution is found in the first 400 iterations:[9]
      i. Update CL' by picking the label position with the lowest objective-function coefficient for each point feature.

[9]This inner loop constitutes the Lagrangian heuristic, with steps (iii) and (iv) constituting the subgradient optimization. Note that the Lagrangian heuristic will be solving relatively simplified versions of the full problem initially, because very few constraints will be included in ACS at first.

ii. Copy CL' to CL if it is better.

iii. If a constraint in ACS is overconstrained (i.e., both conflicting label positions are occupied), the corresponding Lagrangian multiplier is increased, thus increasing the objective-function coefficients for the two label positions involved.

iv. If a constraint in ACS is underconstrained (i.e., both conflicting label positions are not occupied), the corresponding Lagrangian multiplier is decreased, thus decreasing the objective-function coefficients for the two label positions involved.

5. Return CL.

Local minima

If the algorithm were implemented exactly as described above, it would perform quite poorly. The algorithm exhibits two weaknesses: a pronounced sensitivity to local minima, and a tendency to fall into useless cyclic behavior.

To address the worst of these deficiencies, Zoraster recommends a series of modifications to the basic algorithm. The first heuristic he suggests is rescaling the size of the multiplier increments used in 4(c)iii and 4(c)iv. If a specified number of iterations have passed without improving the best solution seen, the algorithm is assumed to be in a region surrounding a local minimum of the objective function. By reducing the multiplier increments periodically, the algorithm is often able to identify improved minima.

In spite of the modifications mentioned above, the algorithm tends to cycle about local minima, constantly re-evaluating a particular sequence of labelings. If two features have overlapping label positions, for example, and both are currently occupied, then the associated objective-function coefficients of both positions will be increased. This will make them less attractive over time and it is likely that both labels will be simultaneously moved to alternate positions. On subsequent iterations, both positions will still overlap but are now unoccupied so their associated coefficients will decrease. This will make both positions relatively more attractive to their respective features and it often occurs that they will be simultaneously reoccupied. This situation is illustrated in Figure 9. In order to avoid this particular type of cyclic behavior, Zoraster discriminates in the overconstrained case, applying the multiplier to only one of the objective-function coefficients; the choice between coefficients is made by examining whether the algorithm is currently in an odd- or even-numbered iteration. This heuristic proves to be crucial to the success of the algorithm but is somewhat disappointing as it has no motivation or analogue in the mathematical formulation.

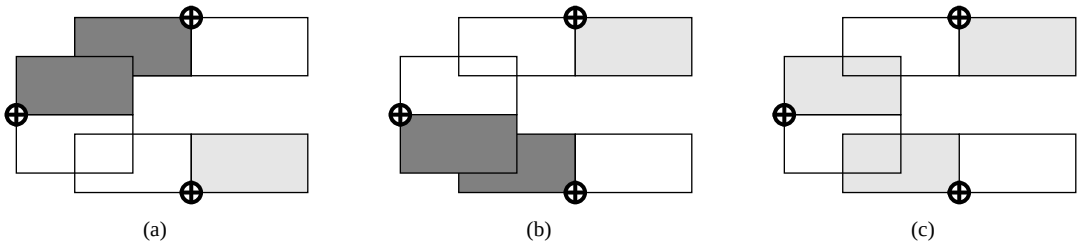


Figure 9: Stable and unstable configurations for Zoraster's approach. The conflict in configuration (a) causes the filled regions of the upper and left points to be disfavored, and the slack in the potential conflict between the lower and left points causes the unfilled regions for those two points to be favored. This leads eventually to modifying the configuration as in (b). This configuration, similarly, eventually leads back to the configuration in (a). The stable configuration (c) is never found.

A more insidious form of cycling can be caused by the intersection of more than two potential label positions. Overplots will gradually be discouraged, yet resolved overplots will result in underconstrained pairwise constraints, which in turn encourage surrounding labels to repopulate the contentious region. This situation is illustrated in Figure 10. Since the center candidate position overplot represents an underconstrained constraint, the left and right labels will be encouraged to move into the conflicted area, despite the fact that this will always introduce a conflict with the top label. As the number of label positions that overlap increases beyond three, the problem is exacerbated since label positionings are encouraged in regions which are often already dense with overplots. Zoraster attempts to address this deficiency by arbitrarily pinning variables (i.e., fixing their values permanently) that are subject to four or more pairwise overplot constraints. If no feasible solution has been identi-

fied after 400 iterations of the Lagrangian heuristic, variables that are subject to more than three overplot constraints are pinned to zero. If after 600 iterations a feasible solution has still not been identified, the current (infeasible) solution is returned to the top level of the algorithm. This is equivalent to arbitrarily eliminating label positions in crowded areas of the map.

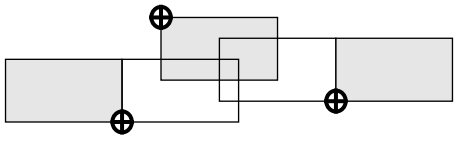


Figure 10: An unstable configuration for Zoraster's algorithm.

Another attempt to control the algorithm's susceptibility to this weakness is the choice of multiplier increments. Zoraster recommends an initial overconstrained stepsize of $+\frac{1}{8}$ and an underconstrained stepsize of $-\frac{1}{16}$. The relative magnitudes of the stepsizes loosely represent the ability of a violated constraint to discourage subsequent reoccupation of a conflicted label position. Although Zoraster offers these values as empirical constants based on his experiments with a variety of different maps, optimal values are probably dependent on the density of the particular labeling problem. Indeed, we obtained better performance by using modified parameter values, as discussed below.

3.7 Implementation of Zoraster's algorithm

Tuning

Our experience with Zoraster's algorithm indicates that some amount of patience is required in determining parameters to achieve optimal performance. Cycling phenomena can be reduced with a few straightforward modifications. In the implementation of the overconstrained heuristic described previously, noticeably better results were achieved by choosing between the objective-function coefficients randomly instead of alternating based on the iteration count. In trials which did not consider cartographic preferences, initial preference weightings were selected randomly as this gave better results than setting them uniformly to zero. These results supported our observation that in general the PFLP algorithms were able to benefit from a reduction of systematicity through an appropriate introduction of randomness.

We also experimented with various values of the multiplier increments and resize factors. Surprisingly, we obtained the best results by setting the underconstrained multiplier decrement to 0, i.e., multiplier values were never decreased. This insured that unobstructed label positions would be favored over time, while competing label positions would still be disfavored in relation to the frequency with which they were overconstrained. Additionally, this alleviated many of the crowding artifacts by focusing the optimization on simply clearing violated constraints. The initial value of the overconstrained multiplier increment was set at $\frac{1}{2}$ as suggested by Zoraster. We achieved slightly improved results using a multiplier resize factor of $\frac{2}{3}$ instead of $\frac{1}{2}$.

Variable pinning

With respect to variable pinning, Zoraster is quite vague, stating only that a small fraction of variables involved in more than three overplot constraints are arbitrarily fixed to zero. We experimented with values of 3, 5, and 10 per cent. When possible, the variables were chosen from different point features in order to avoid deleting features prematurely. Nevertheless, we found the use of variable pinning degraded the quality of labelings tremendously. Since the Lagrangian heuristic is invoked many times over the course of the algorithm, returning an infeasible solution at a particular stage does not necessarily prevent the later identification of feasible solutions. For the point-selection experiments, if the final labeling returned was infeasible, we ran a post-deletion heuristic (described in Section 4) to produce a solution free of overplots. This proved to be far superior to relying on variable pinning to resolve overplots.

3.8 Stochastic search

As we have seen, each of the local search methods can be trapped in local minima of the search space; the inherent intractability of the problem makes this inevitable for any practical algorithm. Nonetheless, we may still hope to improve upon the level of performance exhibited by these algorithms by examining more carefully the frailties that they exhibit.

The problems with the local search methods fall into two classes. First, there are systematic patterns on which the various algorithms get into trouble by getting trapped in local minima. As the number and density of points increases, the odds of seeing these patterns increase correspondingly, and performance may degrade. Second, the particular operations that the algorithms incorporate do not allow for jumping out of a local minimum once one is found. These two behaviors of *systematicity* and *monotonicity* are symptomatic of problems for which stochastic methods tend to work well. Stochastic methods, such as simulated annealing (Kirkpatrick, Gelatt Jr., and Vecchi, 1983; Cerny, 1985) and genetic algorithms (Holland, 1975), attempt to resolve the problems of systematicity and monotonicity by incorporating a probabilistic or stochastic element into the search. Since the stochastic course of behavior is unpredictable, systematic artifacts of the algorithm can be eliminated, and allowance can be made for a suitably limited, nonmonotonic ability to jump out of local minima. It seems natural then to apply a stochastic method to the PFLP problem.

Simulated annealing for PFLP

Simulated annealing (Kirkpatrick, Gelatt Jr., and Vecchi, 1983; Cerny, 1985) is essentially a stochastic gradient descent method that allows movement in directions other than that of the gradient. In fact, the solution is sometimes allowed to get worse rather than better. Of course, such anarchic behavior is not tolerated uniformly. Rather, the ability of the algorithm to degrade the solution is controlled by a parameter *T*, called the temperature, that decreases over time according to an annealing schedule. At zero temperature, such negative steps are disallowed completely, so that the algorithm reduces to a descent method (though not necessarily along the gradient). At higher temperatures, however, a wider range of the space can be explored, so that regions surrounding better local minima (and perhaps even the global minimum) may be visited. The following outline describes the essential characteristics of a simulated annealing algorithm for PFLP:

1. For each point feature, place its label randomly in any of the available potential positions.
2. Repeat until the rate of improvement falls below a given threshold:
   (a) Decrease the temperature, T, according to the annealing schedule.
   (b) Pick a label and move it to a new position.
   (c) Compute ΔE, the change in the objective function caused by repositioning the label.
   (d) If the new labeling is worse, undo the label repositioning with probability $P = 1.0 - e^{-\Delta E/T}$.

The implementation of a standard simulated annealing algorithm involves four components: choice of an initial configuration, an appropriate objective function, a method for generating configuration changes, and an annealing schedule.

Initial configuration. As an alternative to starting with randomly placed labels, one could consider a "piggyback" method where simulated annealing is applied as a post-process to the results of another algorithm. In our experiments, however, this did not lead to either a significantly better solution or faster convergence.

Objective function. The choice of objective function affects the aesthetics of the layout, the quality of the solution, and efficiency of the search. Because simulated annealing is a statistical method which relies on a large number of evaluations for its success, the best objective functions are those for which ΔE can be computed easily. The objective functions we chose counted the number of obstructed labels (if point selection was disallowed) or the number of deleted labels plus the number of obstructed labels. If point selection is allowed, we also considered an objective function which counts the number of pairwise overplots plus the number of deleted labels. This change in objective function doesn't noticeably change the performance of the annealing algorithm, but has the advantage of being significantly faster to compute.

Configuration changes. We have experimented with two strategies for choosing which label to reposition: the label can be chosen randomly from the set of all labels, or it can be chosen randomly from the set of labels that are currently experiencing a conflict. The second method isolates changes to those parts of the map that have conflicts, causing the algorithm to converge faster. When cartographic preferences that distinguish label positions are included in the problem, this simplification is no longer acceptable because the movement of unconflicted labels may affect the current value of the objective function. In our experiments the more time-consuming method of choosing from all available features was used.

Annealing schedule. The initial value of T was selected so that $P = \frac{2}{3}$ when $\Delta E = 1$. At each temperature a maximum of $20n$ labels are repositioned, where n is the number of point features. The temperature is then decreased by 10 per cent. We

employ a Metropolis-style algorithm, always accepting a suggested configuration change if it leads to a lower cost. If more than $5n$ successful configuration changes are made at any temperature, the temperature is immediately decreased. This process is repeated for at most 50 temperature stages. However, if the algorithm stay at a particular temperature for the full $20n$ steps without accepting a single label repositioning, then it stops with the current labeling as the final solution. We found the particular choice of annealing schedule to have a relatively minor affect on the performance of the algorithm as discussed in Section 4. This particular schedule was chosen to provide a reasonable trade-off between efficiency and solution quality; longer annealing schedules result in slightly improved solutions.

4 Comparison Experiments

In order to compare the effectiveness of this wide variety of algorithms for PFLP, we implemented six algorithms chosen from the set of non-exhaustive methods for PFLP. (Our experiments have shown that exhaustive methods are intractable for maps with as few as 50 point features.) The algorithms evaluated included a straw-man random-placement algorithm, in which label positions are assigned in a completely random fashion. This algorithm serves as an effective lower bound on algorithm performance. A greedy algorithm that serves as an efficient variant of the exhaustive methods described in Section 3.1 was also tested. The discrete gradient-descent algorithm was implemented, in addition to the algorithms of Hirsch and Zoraster. Finally, a stochastic algorithm utilizing simulated annealing was implemented. Each of the algorithms (except for Hirsch's) was allowed four candidate placement positions for labels. All candidate positions were taken to be equally desirable, i.e., preferences among different potential label positions were not considered (except where otherwise noted).[10] A complete discussion of the implementation details for all of the algorithms is provided elsewhere (Christensen, 1992).

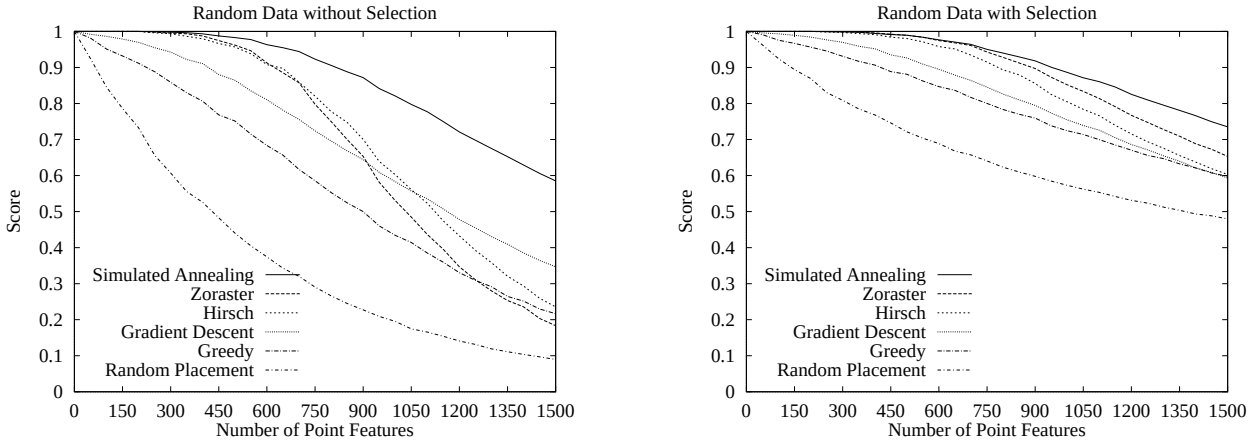


Figure 11: Results of empirical testing of six PFLP algorithms on randomly generated map data with point selection prohibited and allowed.

We began our comparison by testing the performance of each of the algorithms on randomly generated data, with and without point selection, to establish an overall ranking. To determine whether the relative performance of the algorithms was affected by the particular distribution, we then conducted similar tests on naturally occurring point-feature data. Next we ran a series of experiments on two gradient-descent variants in an attempt to improve on the best seen solutions. Finally we investigated the effects of varying the annealing schedule, and noted that the presence of cartographic preferences for candidate positions plays an important role in the usefulness of varying the annealing schedule. For this we conducted four additional trials, comparing the performance of three different annealing schedules while varying the use of point selection as well as the inclusion of cartographic preferences.

In the first group of tests, n point features with fixed-size labels (30 x 7 units) were randomly placed on a grid of size 792 by 612. (These dimensions were selected subjectively in an effort to identify a typical map scale for an 11 by 8.5 inch page size.) Tests were run for n = 50, 100, 150, …, 1500. For each problem size tested, 25 layouts were generated, a score was cal-

[10] In many types of production-quality maps, overplots are often preferred to feature deletion (Ebinger and Goulette, 1990).

culated equal to the fraction of labels placed without overplots, and the results were averaged to give a composite result for the algorithm at that problem size. These tests were then repeated with point selection allowed. For most of the algorithms (greedy, gradient descent, Zoraster, and simulated annealing) this was a natural extension. For the Hirsch algorithm, however, there was no straightforward method of allowing points to be deleted. In order to include Hirsch's algorithm in the point-selection comparisons, we developed a post-pass deletion heuristic which seeks to clear the map of overplots with the fewest number of label deletions possible. This heuristic deletes the feature whose label has the greatest number of conflicts with other (non-deleted) labels. This process is repeated until the map is free from overplots. Although this algorithm is clearly non-optimal (it is straightforward to show that optimal PFLP is reducible to the problem of optimal label deletion and therefore NP-hard), we found it to be an acceptable heuristic in practice. The score was again the fraction of labels placed without conflict. Figure 11 shows the results of these experiments. As these graphs show, simulated annealing performs significantly better across the full range of problems considered. Other perspectives on these results are shown in Figures 12 and 13. Figure 12 shows a particular random map of 750 point features labeled by the six basic algorithms. Figure 13 illustrates the variance across different problem instances for 25 different trials of 750 point features.

Next, cartographic data for Massachusetts were used to test the algorithms on naturally occurring point-feature distributions obtained from the GNIS state file for Massachusetts (United States Geological Survey 1990). The algorithms were again scored based on the number of unconflicted labels, both with and without point selection. At each problem size, 25 layouts were generated by choosing randomly from the data file. For example at $n = 350$, each problem instance was generated by choosing 350 point features randomly from the GNIS data. Tests were run for $n = 50, 100, 150, \ldots, 500$. Figure 14 shows the results of these tests. Because the ratio of average label size to available map area is significantly larger for the Massachusetts examples, and due to clustering of the point features, the performance of the algorithms deteriorates faster in the graphs of Figure 14 relative to Figure 11. Nonetheless, the overall rankings were preserved.

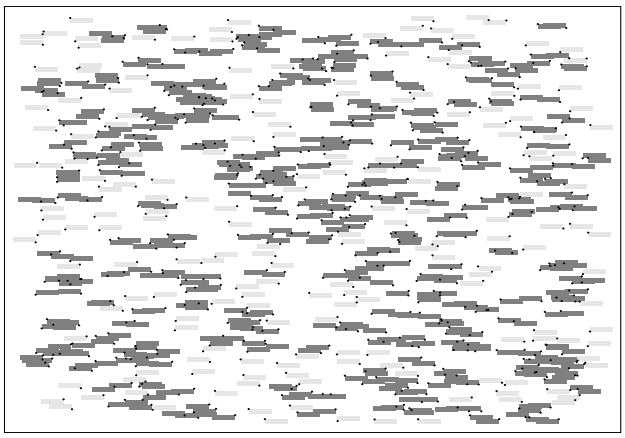
Though the simulated annealing algorithm easily dominated the competing algorithms, we noted that the discrete gradient-descent algorithm performed surprisingly well, especially at high densities, given its simplicity. To investigate the promise of this approach in more detail, we implemented two related algorithms, "2-opt" and "3-opt" discrete gradient-descent algorithms which consider the best sequence of two and three repositionings at each iteration.[11] A practical implementation of these algorithms is moderately complicated and requires a careful strategy for selective rescoring of repositionings at each iteration, supporting data structures for efficient search of a table of repositionings, and some clever record-keeping measures. Figure 11 shows the results of these new variants compared with the original discrete gradient-descent algorithm, the simulated annealing algorithm and the random placement algorithm. Although the "2-opt" and "3-opt" algorithms each improve on the performance of their predecessor, the degree of improvement grows less in each case, hinting towards an asymptote around the performance of the simulated annealing algorithm. Further, even with a very careful implementation, the computational requirements of the 2-opt and 3-opt algorithms quickly become unreasonable as the number of candidate positions increases.

The next set of experiments investigated the effect of the annealing schedule on the performance of the simulated annealing algorithm. We found that for very simple objective functions, e.g., the original 4-position model without placement preferences, most potential label repositionings have no effect on the value of the objective function. For such spaces, a simple random descent (the equivalent of zero-temperature simulated annealing) performs nearly as well as simulated annealing at medium and even long schedules. This is seen in Figure 14. As the terrain of the search space becomes rougher, and involves a greater number of local minima, the utility of the annealing schedule is increased. Figure 14 shows that in experiments involving a 4-position model with placement preferences, the performance of zero-temperature annealing drops roughly to that of the discrete gradient-descent algorithm.[12]

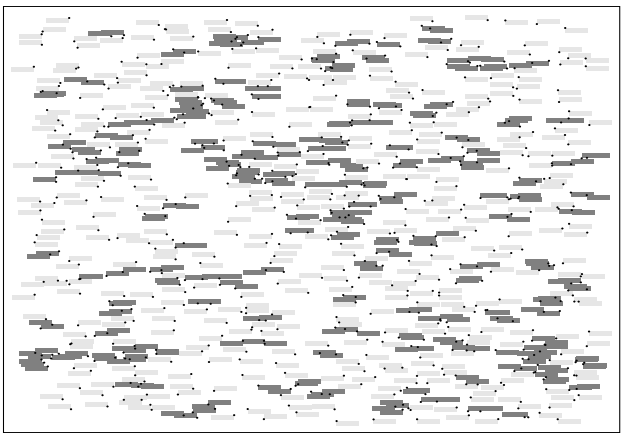
Computational resources required for the various algorithms vary dramatically, but not unexpectedly. As a rough indication of algorithm performance, Figure 18 depicts a scatterplot of running times for each of the algorithms running on a DEC 3000/400 AXP workstation. To the extent that these running times are representative of the intrinsic computational require-

[11]We use these terms because of the similarity of these methods to the *k*-opt methods proposed for the NP-complete Traveling Salesman Problem (TSP). Variants of this method comprise the current best methods for the TSP (Johnson, 1990).

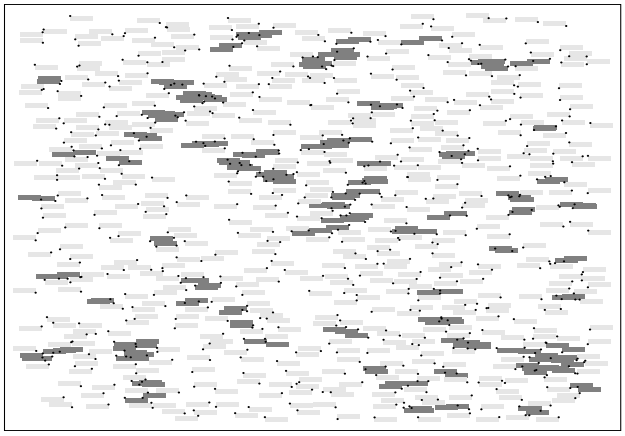
[12]Note that the performance of the gradient-descent algorithm appears to have increased relative to the original experiments. Because the original objective function yields a search space with many flat plateaus, the algorithm is often unable to find the edge of a plateau and terminates; the modified objective function yields virtually no plateaus and the algorithm is able to continue further before reaching a local minimum. A second reason for the improvement is the inclusion of preferences in the score metric. Since the score considers a larger dynamic range, the scale of the graph along the y-axis is more compressed, resulting in a closer grouping of the algorithms. (Notice the relatively higher performance of random placement as compared with the previous trials.)



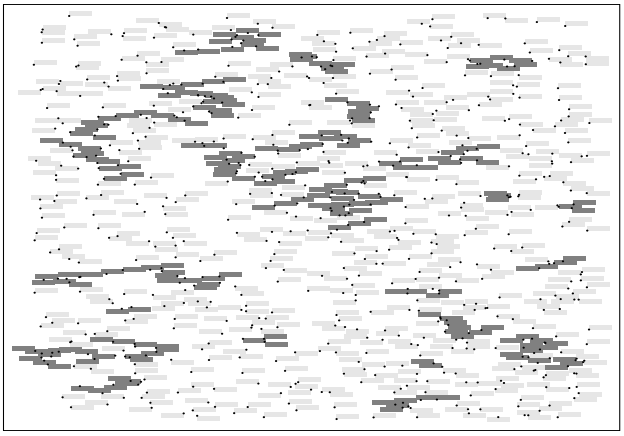
Random Placement (564)



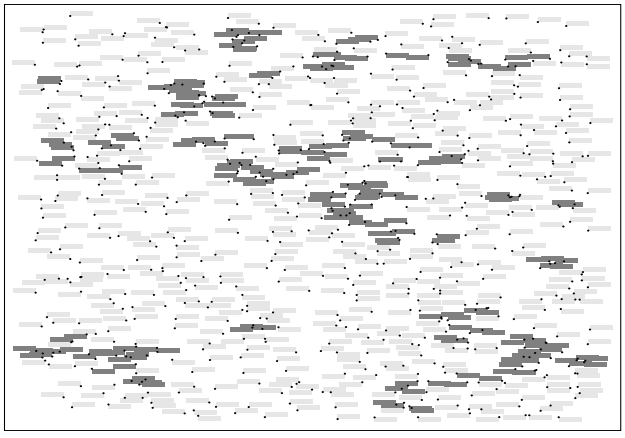
Greedy Depth-First Placement (341)



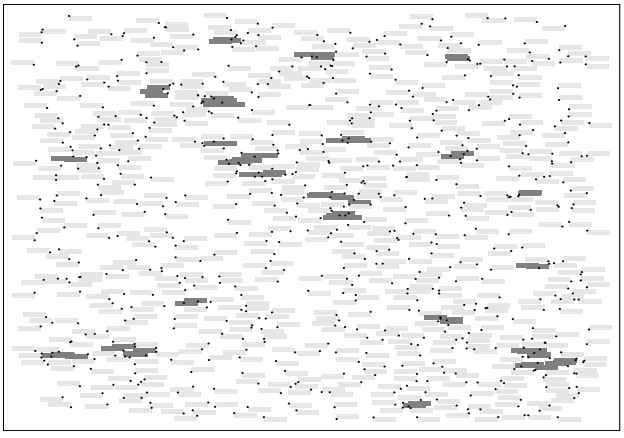
Discrete Gradient Descent (222)



Hirsch's Algorithm (222)



Zoraster's Algorithm (219)



Simulated Annealing (75)

Figure 12: A sample map of 750 point features with labels placed by the six different algorithms. Labels printed in dark grey overplot other labels or points. Labels printed in light gray are free of overplots. Numbers in parenthesis indicate the final value of the objective function computed as the number of labels with overplots.

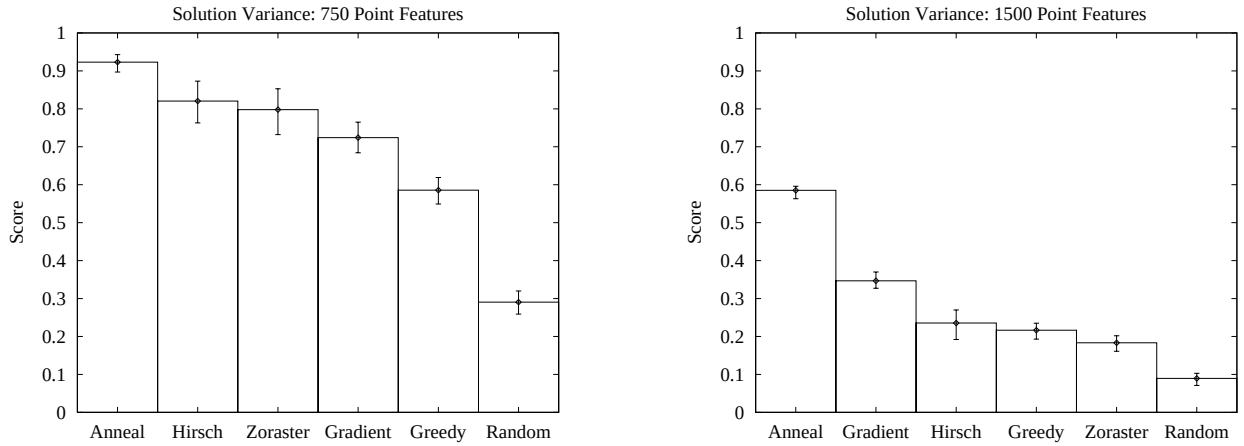


Figure 13: Range of results generated for 25 different labeling problems involving 750 and 1500 point features. The worst case of simulated annealing falls significantly above the best case of competing algorithms, even across different trials.

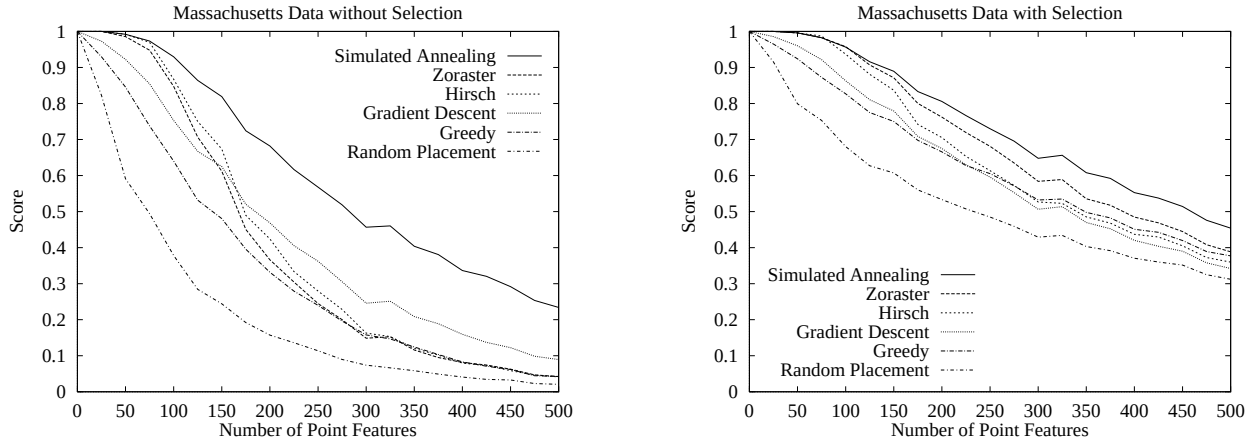


Figure 14: Results of empirical testing of six PFLP algorithms on GNIS data for Massachusetts with point selection prohibited and allowed.

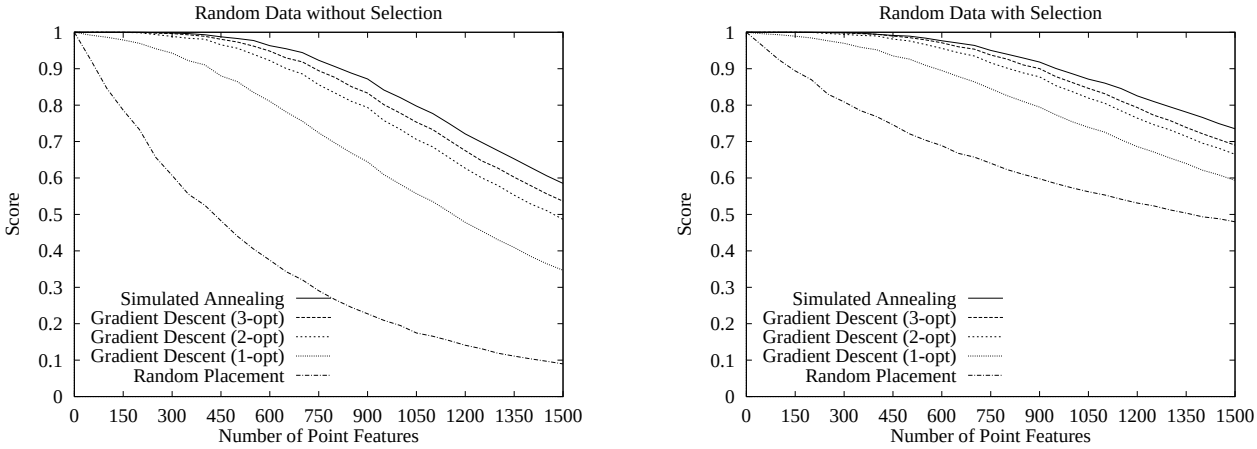


Figure 15: Results of empirical testing of discrete gradient-descent algorithms on randomly generated map data.

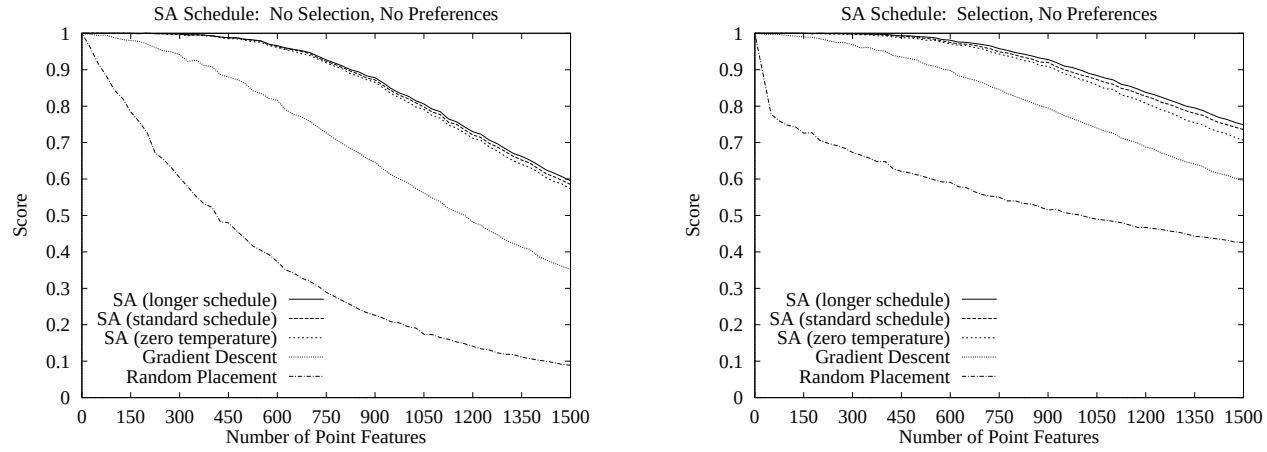


Figure 16: Comparison of annealing schedules against a gradient-descent algorithm without cartographic preferences.

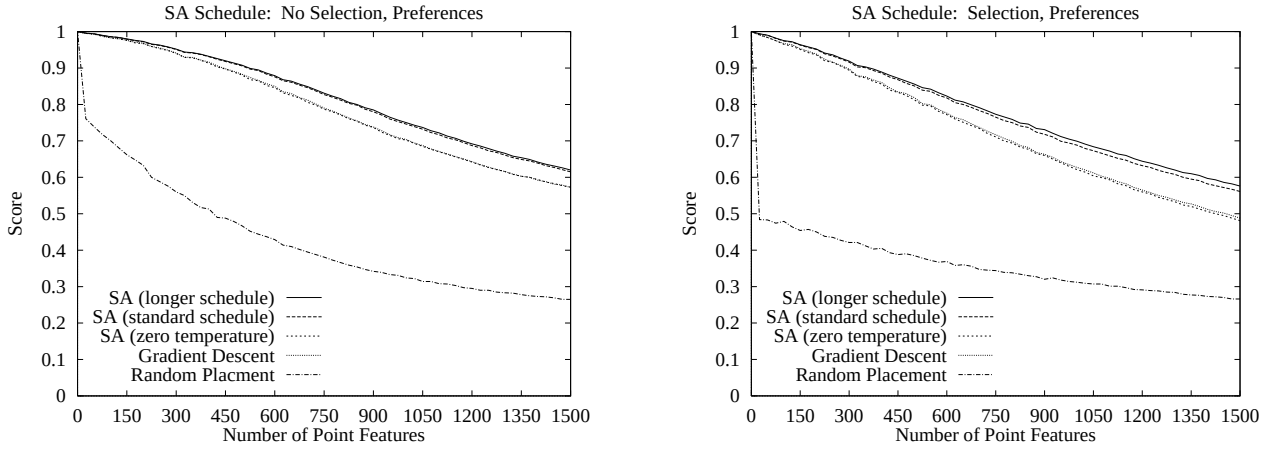


Figure 17: Comparison of annealing schedules against a gradient-descent algorithm with cartographic preferences. (Note that the lines corresponding to zero-temperature annealing and gradient descent lie very close together).

ments of each algorithm, certain subsumption relationships can be derived. In Figure 18a, for example, Zoraster's algorithm lies to the lower right of the 3-opt discrete gradient-descent algorithm, indicating that it is both slower and exhibits inferior solutions. The 3-opt algorithm, in turn, is dominated by the simulated annealing algorithm. Eliminating algorithms which are subsumed by the two algorithms leaves a "staircase" of algorithms which, depending on time vs. solution quality requirements, would be preferred for a given task. At both densities shown, this staircase includes, in order of increased computation time and solution quality: random placement, the greedy algorithm, the original gradient-descent algorithm, the 2-opt gradient-descent algorithm, and the simulated annealing algorithm.

5 Conclusions

The point-feature-label placement problem is a graphics-design problem of practical importance and noted difficulty. Analysis of the computational complexity of the problem bears out its inherent difficulty; the search for good heuristic solutions thus becomes important. In this paper, we have proposed two new algorithms for PFLP — variants of discrete gradient descent and simulated annealing — for PFLP, and compared them with previously proposed algorithms. This empirical testing, which constitutes the first such comparative study, provides the basis for a graphic comparison of the time-quality tradeoff in label-placement algorithms, demonstrating that certain algorithms — 3-opt gradient descent, Zoraster's, and Hirsch's algorithm, for instance — are subsumed by others in both speed and quality. The experiments also argue for the use of simulated annealing

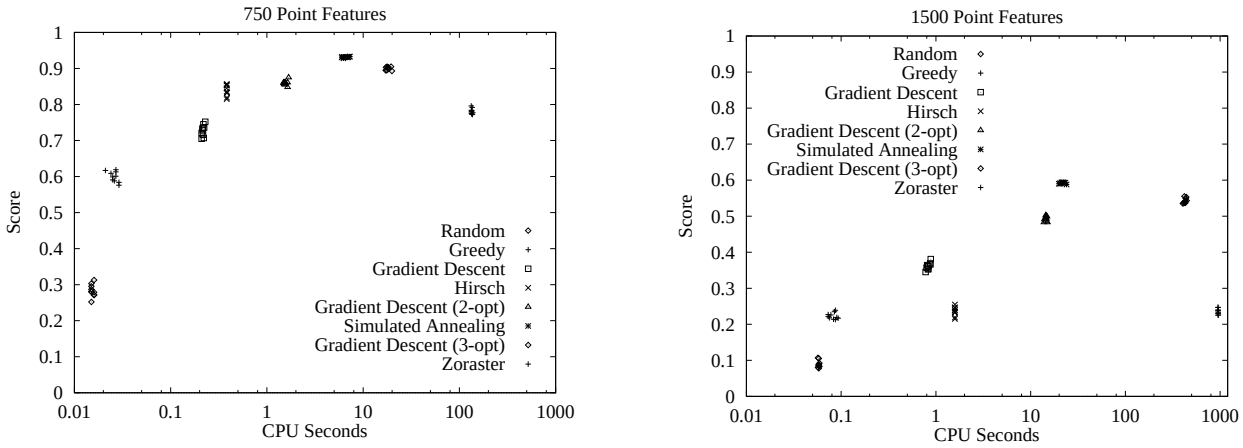


Figure 18: Running times for 10 different trials of 750 point features and 1500 point features (note the logarithmic scale of the x axis).

over the alternatives when overall solution quality is critical. For time-critical applications, the annealing schedule can often be shortened or eliminated altogether while still providing reasonable solutions. This result stands in contrast to previous empirical investigations of simulated annealing, which have shown that for a few NP-hard problems simulated annealing is competitive with customized heuristic techniques, but typically only when allowed to run for very long periods of time (Johnson et al., 1989; 1991). Simulated annealing has the additional advantage of being one of the easiest algorithms to implement. Table 1 gives the number of lines of code for each of the algorithms under our implementation, as an admittedly rough indication of implementation complexity.[13]

Algorithm	Lines of C code
Random Placement	20
Greedy	79
Gradient Descent (1-opt)	210
Simulated Annealing	239
Zoraster	346
Hirsch	381
Gradient Descent (2-opt)	1807
Gradient Descent (3-opt)	2284

Table 1: Lines of source code for label placement algorithms

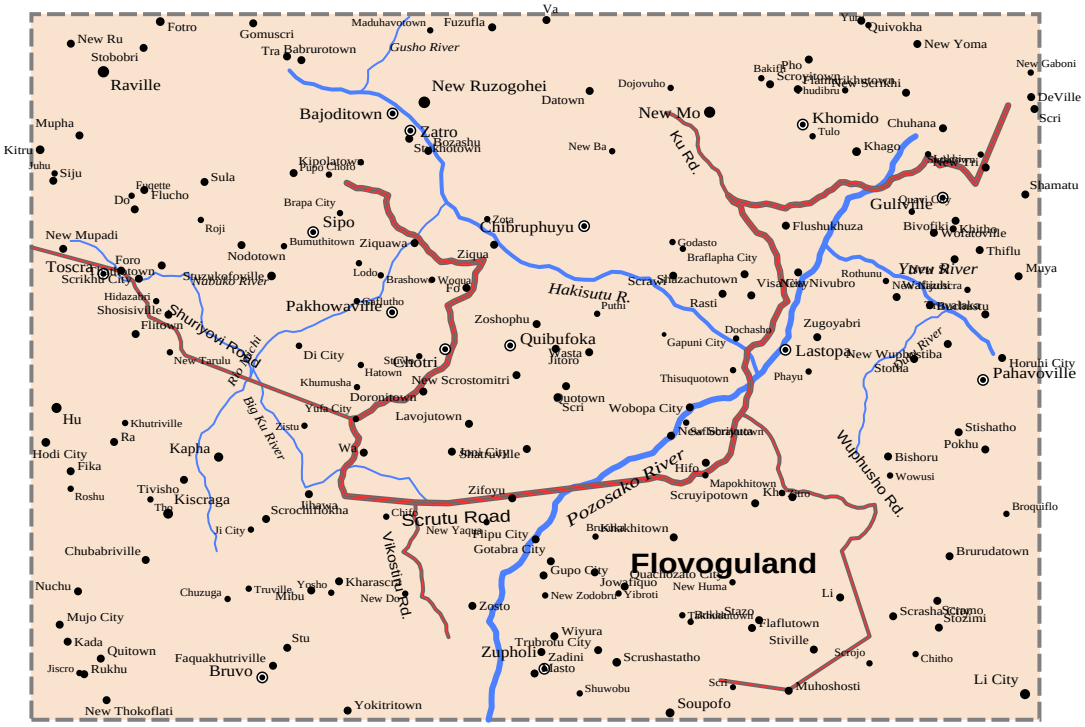
Unlike much of the previous work on label placement, the approach we have suggested cleanly separates the combinatorial-optimization aspect of the problem from the candidate-position modeling aspect. This way of stating the problem allows for the search algorithms discussed here to be used with more advanced cartographic positioning models. Modifying the algorithm to generate new sets of potential label positions, which is necessary to permit the labeling of line and area features, is accomplished easily, provided adequate models of line-feature (Ebinger and Goulette, 1990) and area-feature labeling (Carstensen, 1987; van Roessel, 1989) are available. Figure 19 shows a sample map involving all three feature types, as

[13]Our implementation makes extensive use of function pointers to provide dynamic reconfiguration of the basic aspects of each algorithm. As a result, however, these numbers are undoubtedly higher than those which would occur in more straightforward implementations.

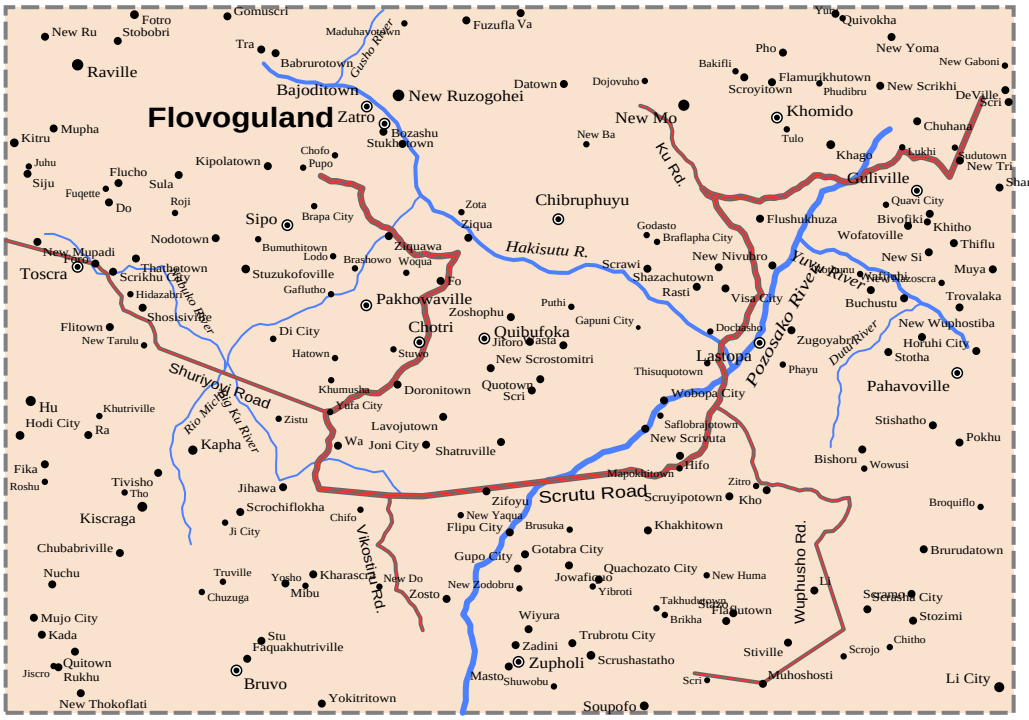
labeled by the simulated annealing algorithm (Edmonds et al., 1994). Changing the objective function to allow for a priori placement preferences, sophisticated point selection, and complex interactions between labels and map symbology is also possible.

6 Acknowledgments

The research reported in this paper was funded in part by a contract with U S WEST Advanced Technologies, by Presidential Young Investigator Award IRI-9157996 from the National Science Foundation, and by a grant from Digital Equipment Corporation. Andy Breeding, an information analyst at Digital Equipment Corporation, assisted us in the compilation of the bibliography. Thanks also to Tom Ngo and Shawn Edmondson for additional support.

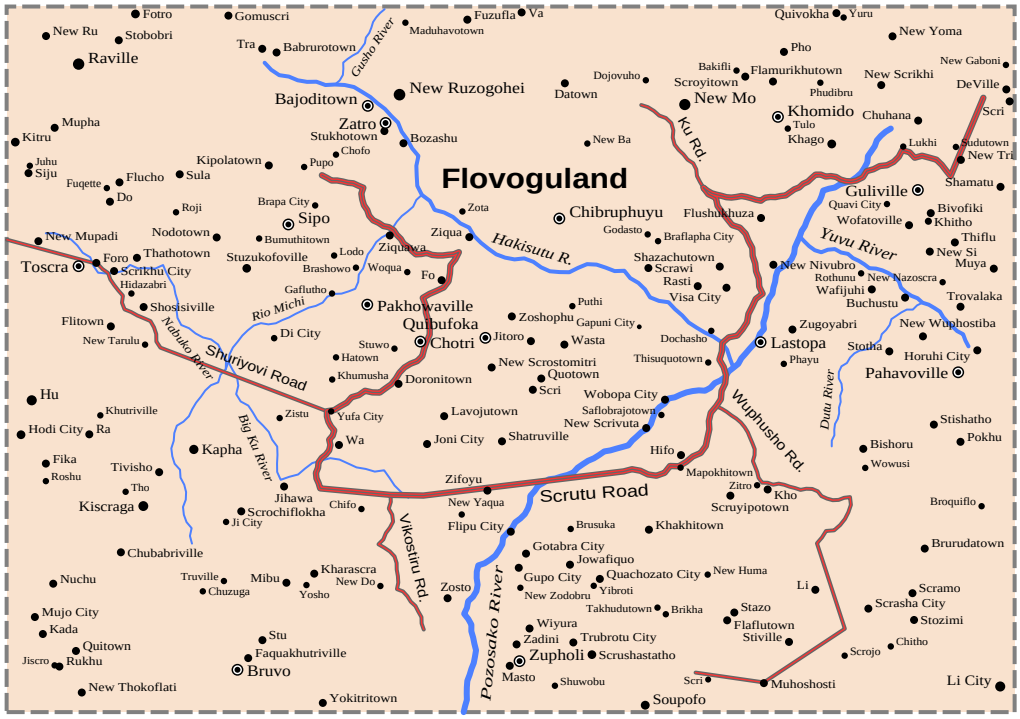


(a)



(b)

Figure 19: A map involving line, area, and point features labeled by the simulated annealing algorithm. The initial random labeling is shown in (a). An intermediate configuration of the algorithm is shown in (b). The final labeling is shown in (c).



(c)

Figure 19: A map involving line, area, and point features labeled by the simulated annealing algorithm. The initial random labeling is shown in (a). An intermediate configuration of the algorithm is shown in (b). The final labeling is shown in (c).

References

Ahn, J. and H. Freeman. 1984. A program for automatic name placement. *Cartographica*, 21(2&3):101-109, Summer & Autumn. Originally published in *Proceedings of the Sixth International Symposium on Automated Cartography (Auto-Carto Six)*, Ottawa/Hull, October 1983.

Carstensen, L. W. 1987. A comparison of simple mathematical approaches to the placement of spot symbols. *Cartographica*,24(3):46-63.

Cerny, V. 1985. A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41-51.

Christensen, J., J. Marks, and S. Shieber. 1992. Labeling Point Features on Maps and Diagrams. Center for Research in Computing Technology, Harvard University, TR-25-92, Dec.

Christensen, J., J. Marks, and S. Shieber. 1993. Algorithms for Cartographic Label Placement. *Proceedings of the American Congress on Surveying and Mapping '93*, Feb.

Christensen, J., J. Marks, and S. Shieber. 1994. Placing Text Labels on Maps and Diagrams. *Graphics Gems IV.* Academic Press, pages 497-504.

Cook, A. C. and C. B. Jones. 1990. A Prolog rule-based system for cartographic name placement. *Computer Graphics Forum*, 9:109-126.

Consorti, V., L.P. Cordella, and M. Iaccarino. 1993. Automatic lettering of cadastral maps. *Proceedings of the International Conference on Document Analysis and Recognition*, page 129-132, Tsukuba Science City, Japan, October.

Cromley, R. G. 1986. A spatial allocation analysis of the point annotation problem. In *Proceedings of the Second International Symposium on Spatial Data Handling*, pages 38-49, Seattle, Washington, July. International Geographical Union and International Cartographic Association.

Dechter, R. and J. Pearl. 1985. Generalized best-first search strategies and the optimality of A*. *Journal of the Association of Computing Machinery*, 32(3):505-536.

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

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

Edmondson, S., J. Christensen, J. Marks, and S. Shieber. 1994. A General Cartographic Labeling Algorithm. *In preparation*.

Fisher, M. L. 1981. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27:1-18.

Formann, M. and F. Wagner. 1991. A packing problem with applications to lettering of maps. In *Proceedings of the Seventh Annual Symposium on Computational Geometry*, pages 281-288, North Conway, New Hampshire, July. ACM.

Freeman, H. and J. Ahn. 1987. On the problem of placing names in a geographic map. *International Journal of Pattern Recognition and Artificial Intelligence*, 1(1):121-140.

Freeman, H. 1988. An Expert System for the Automatic Placement of Names on a Geographical Map. *Information Sciences*, 45:367-378.

Freuder, E. C. 1982. A sufficient condition for backtrack-free search. *Journal of the Association of Computing Machinery*, 29(1):24-32.

Garey, M. R. and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, New York.

Gaschnig, J. 1979. *Performance Measurement and Analysis of Certain Search Algorithms*. Ph.D. thesis, Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Haralick, R. M. and G. L. Elliot. 1980. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263-313.

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

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

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

Imhof, E. 1975. Positioning names on maps. *The American Cartographer*, 2(2):128-144.

Johnson, D. S. 1990. Local optimization and the traveling salesman problem. In *Proceedings of the 17th Colloqium on Automata, Languages, and Programming*, pages 446-461.Springer-Verlag.

Johnson, D. S., C. R. Aragon, L. A. McGeoch, and C. Schevon. 1989. Optimization by simulated annealing: An experimental evaluation; part I, graph partitioning. *Operations Research*, 37(6):865-892.

Johnson, D. S., C. R. Aragon, L. A. McGeoch, and C. Schevon. 1991. Optimization by simulated annealing: An experimental evaluation; part II, graph coloring and number partitioning. *Operations Research*, 39(3):378-406.

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

Karp, R. M. 1972. Reducibility among combinatorial problems. *Complexity of Computer Computations*. Plenum Pres, New York, pages 85-103.

Kato, T. and H. Imai. 1988. The NP-completeness of the character placement problem of 2 or 3 degrees of freedom. *Record of Joint Conference of Electrical and Electronic Engineers in Kyushu*, 1138. In Japanese.

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

Korf, R. E. 1988. Search: A survey of recent results. In H. E. Shrobe, editor, *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*. Morgan Kaufmann, San Mateo, California, pages 197-237.

Langran, G. E. and T. 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.

Marks, J. and S. Shieber. 1991. The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard University, March.

Mower, J. E. 1986. Name placement of point features through constraint propagation. In *Proceedings of the Second International Symposium on Spatial Data Handling*, pages 65-73, Seattle, Washington, July. International Geographical Union and International Cartographic Association.

Noma, E. 1987. Heuristic method for label placement in scatterplots. *Psychometrika*, 52(3):463-468.

Papadimitriou, C. H. and K. Steiglitz. 1982. *Combinatorial Optimization: Algorithms and Complexity.* Prentice Hall, Englewood Cliffs, New Jersey.

Purdom, P. W. 1983. Search rearrangement backtracking and polynomial average time. *Artificial Intelligence*, 21(1,2):117-133.

Sahni, S. 1974. Computationally related problems. *SIAM Journal of Computing*, 3:262-279.

United States Geological Survey, National Mapping Division. 1990. Geographic Names Information System, November.

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

Wu, C. V. and B. P. Buttenfield. 1991. Reconsidering rules for point-feature name placement. *Cartographica*, 28(1):10-27, Spring.

Yoeli, P. 1972. The logic of automated map lettering. *The Cartographic Journal*, 9(2):99-108, December.

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

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

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