Practical Results Using Simulated Annealing for Point Feature Label Placement

Steven Zoraster

Practical Results Using Simulated Annealing for Point Feature Label Placement

Steven Zoraster

ABSTRACT. This paper describes a production-oriented computer program for point feature label placement on petroleum industry basemaps. The program uses a simulated annealing algorithm to find a feasible and near optimal solution to the problem of placing point feature labels on a map near their corresponding symbols, without overplotting each other or other point feature symbols. The program implementation extends recently published work on the use of stochastic algorithms to solve the label placement problem. Results show that simulated annealing provides an efficient and robust solution to map labeling problems for the petroleum industry.

KEYWORDS: automated cartography, automated map design, map lettering, text placement, combinatorial optimization, simulated annealing

Introduction

Recent research has confirmed the feasibility of using stochastic optimization algorithms to solve map label placement problems (Christensen et al. 1995, Edmondson et al. 1997). But that research did not address some important issues encountered in production-oriented mapping environments. This paper explores the use of a stochastic optimization algorithm called "simulated annealing" to find feasible and near optimal solutions to point feature label placement (PFLP) problems encountered in mapping for the petroleum industry. The result of this effort is a new program for map label placement.

The new program was developed to eliminate label-label and label-symbol overplotting on petroleum industry basemaps. Over one hundred petroleum companies around the world use these maps to display the locations and attributes of wells and seismic surveys. Most such maps are computer generated. Tens of thousands of these maps may be generated by a single company each year, and a single map may have thousands of well labels and more than 10,000 seismic labels. The label placement problems in the maps are characterized by:

- significant feature clustering,
- multi-line labels, and
- three types of point feature labels.

Steven Zoraster is a researcher at Landmark Graphics, 220 Foremost Drive, Austin, Texas, 78745. E-mail: <szz@zycor.lgc.com>.

The program described in this paper provides very good results for map label placement. For examples, see Figures 1 and 2.

Figure 1 shows the solution to a randomly generated PFLP problem with 250 individual wells and 24 positions for each well label. One hundred and ninety-four labels were placed without overplotting in 19 CPU seconds on a SGI/Extreme (note 1). Figure 2 shows 20 well symbols and well labels randomly positioned in a tight cluster. For this example, 15 labels were placed in less than 1.5 CPU seconds.

The remainder of this paper is organized as follows. The map label placement problem is discussed in the second section, followed by a section on the use of simulated annealing to solve general combinatorial optimization problems. The fourth section describes petroleum industry basemaps and the types of point features posted on those maps. In the fifth section the label placement program, which is the main focus of this paper, is presented. Results of using the program on real data are presented in the sixth section. Next is a discussion of the results, followed by conclusions.

The Map Label Placement Problem

A major problem in computer-generated maps is overplotting between map labels and between map labels and map symbols. Map label placement has been estimated to take 50% of the time required to compile a map by hand (Cook and Jones 1990).

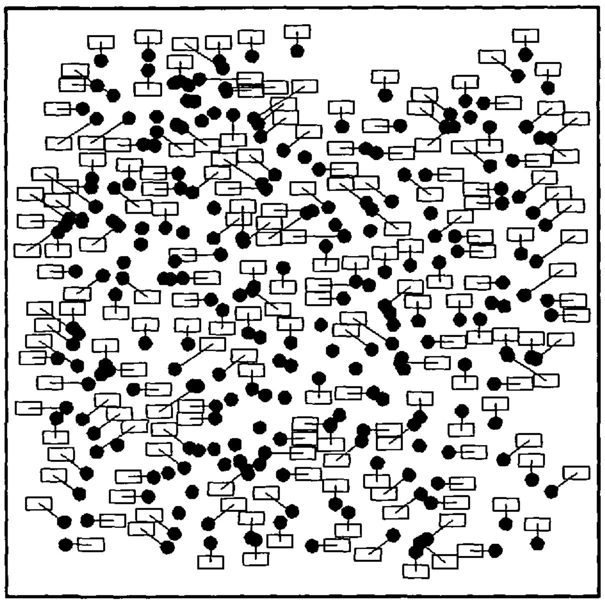

Figure 1. Overplot resolution of a randomly generated label placement problem with 250 point features and 24 possible positions for each label around its feature. Solution time was 19 CPU seconds. Fifty-six labels were deleted from the map to achieve a problem solution.

Over the last 15 years many attempts have been made to solve the map label placement problem automatically. Some algorithms focused on placing individual labels for cartographic line or area features such as rivers or political units (Ahn 1984, pp. 37-51, van Roessel 1989). Most research has focused on resolving label overplotting by moving the labels. Most published algorithms have relied on rule-based implementations which follow depth-first search procedures to explore label placement options (Ahn and Freeman 1984, Mower 1989, Doerschler and Freeman 1992, Cook and Jones 1990). An early paper by Hirsch (1982) suggested using a purely mathematical approach—continuous gradient-descent—but that work was ignored by most other researchers.

Cromley (1985) and Zoraster (1991) suggested an alternative—and arguably better—approach to the problem. They both asserted that map label placement was an exercise in combinatorial optimization and that it should be solved as such. In other words, the label placement problem should be formulated with an objective function to be minimized (e.g., the number of labels deleted from the map), subject to a set of constraints (e.g., no label-label and label-symbol overplotting), and solved using mathematical optimization algorithms. Cromley provided a simple example supporting the use of linear programming to resolve label overplotting. Zoraster (1990) implemented a PFLP algorithm based on a 0-1 integer programming formulation of the problem, which was solved through Lagrangian relaxation and subgradient optimization.

Christensen et al (1995) implemented a general program for PFLP to compare the performance of several mathematical optimization algorithms. The algorithms included a version of Zoraster's integer programming algorithm, a version of Hirsch's continuous gradient-descent algorithm, a discrete gradient-descent algorithm, and a stochastic optimization algorithm using simulated annealing.

Interest in stochastic optimization for improved map label placement was probably motivated by the use of stochastic algorithms to solve a somewhat similar computer-aided design problem: resolving space conflicts in laying out component features on computer chips (Siarry et al. 1987). However, interest in stochastic optimization was inevitable, once the label placement problem was recognized as a hard combinatorial optimization problem, and it has been shown that this particular problem is a NP-hard optimization problem (Formann and Wagner 1991, Marks and Shieber 1991). A polynomial time solution to this type of problem is unlikely to exist (Nemhauser and Wolsey 1988, pp. 138). Useful—but not necessarily optimal—solutions to NP-hard problems are often identified using stochastic algorithms.

Christensen et al. (1995) compared the various label placement algorithms in their program on randomly generated maps with respect to complexity of implementation, computational time, and quality of results measured by the number of labels deleted from the map. In quality of results, simulated annealing generally dominated the other algorithms, while discrete gradient-descent ran fastest and was easiest to implement.

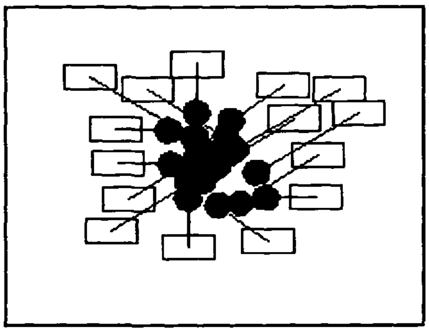

Figure 2. An extreme example of the solution to a label placement problem with densely clustered point features. There are 20 randomly placed point features on this map and 15 of them have been successfully labeled in less than 1.5 CPU seconds. Twenty-four positions were allowed for each label. Multiple runs of this problem produced results with between seven and four deleted labels, all solved in less than 2 CPU seconds.

The maps in Christensen et al. (1995) show extremely high label densities, but are unrealistic in some other ways. For example, they allow only four positions for each point feature label, show only one type of label, and make no distinction between the comparative merit of different label positions relative to the feature symbol.

Another possible criticism of the work by Christensen et al. was that it applied only to point features. Edmondson et al. (1997), working under the direction of Shieber, addressed this potential complaint by generalizing their results to include area and line features. In Edmondson's paper many positions are allowed for each feature label (19 for each point feature label), but the examples presented do not represent high-label density.

Practical mapping problems may contain very dense labeling, multi-line labels, multiple types of labels, many possible positions for some labels, feature clustering, and other characteristics which were not tested comprehensively by Christensen et al. (1995) or Edmondson et al. (1997). This paper builds on their work to explore these issues using simulated annealing to resolve point feature label overplotting.

Simulated Annealing

According to a definition in *The American College Dictionary* (Barnhart 1962) to anneal is "to free from internal stress by heating and gradually cooling." In the world of mathematical optimization, simulated annealing is an iterative and stochastic optimization technique used to minimize an objective function subject to constraints by a sequence of steps modeled on the physical annealing process (Kirkpatrick et al. 1983). In the physical analogy, the objective function is the total energy of a crystal, E, which is minimized by the formation of a perfect crystal. Simulated annealing is often applied to optimization problems where a continuous gradient function to guide the minimization process does not exist, or where standard gradient following algorithms are likely to become trapped at a local minimum. Many of the problems to which simulated annealing is successfully applied are NP-hard problems.

At each iteration a simulated annealing algorithm randomly perturbs the state of a system and the corresponding existing candidate solution to obtain a new system state and a new candidate solution. The algorithm always accepts changes in the system state which decrease or do not change the objective junction value. It also accepts some changes that increase it. The latter are accepted with probability

$$p = e^{(-\Delta/T)} \qquad (1)$$

where

Δ = the change in the objective function value, and

T = a control parameter.

By analogy with the physical model, T is known as the system temperature. The temperature is set to a high value at the start of the optimization process and slowly reduced during its course according to some schedule (hence "annealing"). High temperatures correspond to high probabilities of accepting "bad," non-improving moves (p approaches 1, as T approaches ∞, for any positive Δ). Conversely, very low temperatures correspond to arbitrarily low probabilities of accepting non-improving moves (p approaches 0, as T approaches 0, for any positive Δ). The ability to accept non-improving changes in the objective function is the feature that allows simulated annealing to escape from local minima.

Identification of a global optimum by simulated annealing is statistically guaranteed, provided that the temperature reductions are "small enough," and that for each temperature the number of perturbations is "large enough." In fact, most practical applications settle for near optimal solutions and make corresponding compromises in the annealing schedule.

There are two key issues involved in developing a simulated annealing-based algorithm. One is defining an objective function which:

- Incorporates problem constraints that are handled implicitly in simulated annealing instead of explicitly as in many other optimization algorithms; and
- Allows changes in the objective function value due to perturbations of the system to be evaluated quickly.

The other issue is setting the annealing schedule, including:

- Selecting the initial value for T;
- Deciding how often and how much to reduce T; and
- Deciding how many perturbations to allow in for each temperature.

The objective function for the program presented in this paper is described below, as is the annealing schedule used.

For the map label placement problem, the state of the system encompasses the positions of all the labels. Christensen et al. (1995) developed an objective function for the map label placement problem which incorporates label conflicts so that the influence on the objective function value of moving a single label could be calculated quickly. I borrowed their annealing schedule and a modified version of their objective function for use on petroleum industry basemaps.

Petroleum Industry Basemaps

As mentioned earlier, petroleum industry basemaps are compiled to show the locations and attributes of wells and seismic surveys. Map scales may vary from 1:10,000 to 1:200,000. The maps have three basic types of point feature labels: well labels, seismic shot point labels, and seismic line name labels. Because there is so much information posted on a petroleum industry basemap, it is permitted to resolve overplotting by deleting some labels from the map.

Figures 3a and 3b show a simulated seismic survey before and after the label overplot problem was resolved. For cartographic purposes, a seismic survey can be considered a set of 2D lines. Each individual seismic line is a polyline with point features at each vertex. The point features are shot point locations, or "shot points," where seismic energy reflected off subsurface rocks has been recorded at an array of receivers that were embedded in the ground or towed behind a survey ship.

The usual information posted at seismic shot points is the time required for the seismic energy to travel from an energy source near the surface

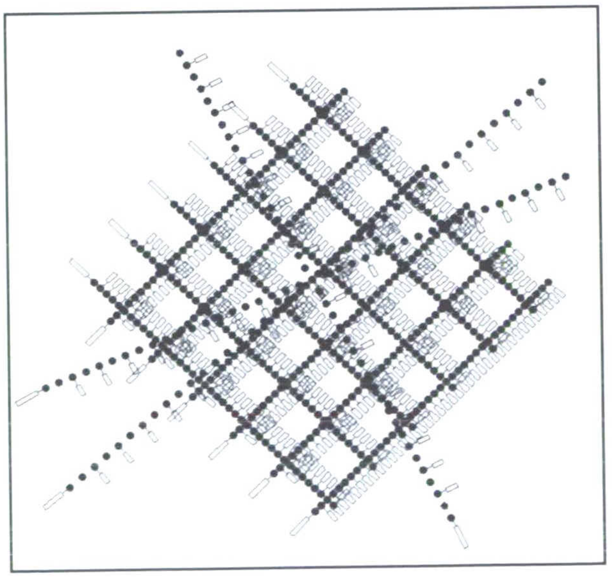

Figure 3a. A simulated seismic label placement problem with 600 shot points and 540 shot point and line name labels before label overplot resolution.

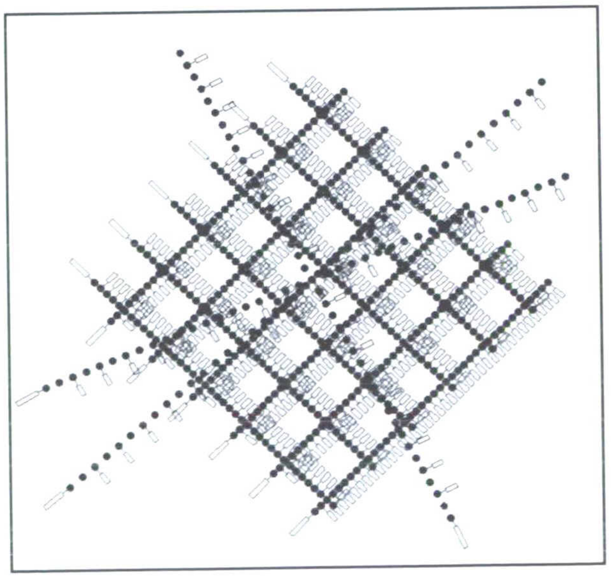

Figure 3b. A simulated seismic label placement problem with 600 symbols and 540 labels after label overplot resolution. 184 labels were deleted from the map. Total CPU time was less than 2 CPU seconds.

down to a particular reflecting rock structure and then back up to the shot point. Shot point labels are oriented perpendicularly to the local trend of a seismic line. However, not all shot points must be labeled. Reflection times may be posted at every Nth shot point, or sometimes not posted at all. The two standard techniques for resolving shot point label conflicts are to 1) move a label to the opposite side of the line, or 2) delete the label from the map.

Each seismic line has one or two line name labels associated with it. Line names are posted at the ends of the line and oriented along the general trend of the line. To prevent overplotting, only one line name needs to be posted for a given line. Sometimes only one line name should be posted for any line. Resolving line name overplots without deletion usually involves rotating the label 30 degrees off the trend of the seismic line. There may be as few as one seismic line on a basemap and as many as 500. On a basemap with many seismic lines there may be 20,000 shot points. On a small-scale regional basemap there may be several areas where many seismic lines have been shot, surrounded by large empty regions where no data is posted.

The third type of point feature on petroleum industry basemaps marks a well location. The well location symbol is posted at the top of the well where the drilling equipment was located or at the bottom of the well path, which is likely to be offset from the top.

The preferred positions for labels around a well symbol are similar to the eight choices commonly discussed in the cartographic literature (Imhof 1975). In general, positions above and to the right of the symbol are preferred. However, like seismic lines, well symbols may be clustered. An oil field in the North Sea or the Gulf of Mexico may have 20 or 30 wells drilled from a single platform. Consequently, more labels are positioned far away from their symbol on this type of map than on more traditional maps. The general pattern for the 24 positions allowed in the program described in this paper are shown in Figure 4. The program determines exact offsets as a function of label and symbol size. Although positions far away from the symbol are numbered so that positions above and to the right of the symbol are generally preferred, this is not meant to define a carefully considered, "cartographically correct" pattern. Once a label moves far away from its symbol, it is basically just fighting for space on the map.

Program Implementation

The program goal is to identify a feasible and near optimal placement of labels on a map, with no pair of labels overplotting each other and with no label overplotting another point feature symbol. The work performed by Christensen et al. (1995) provided the motivation for my implementation. I made modifications and extensions to handle the characteristics of petroleum industry basemaps.

Problem Objective Function

A critical step in the formulation of a problem that is to be solved by simulated annealing is to design the problem objective function. For each label, the label position and overplots with other labels and map symbols contribute to the "cost" incorporated in the objective function to be minimized. The problem objective function incorporates four cost factors:

- Label movement away from the "optimal" position;
- Label deletion;
- Label-symbol conflicts; and
- Label-label conflicts.

These factors are implemented to emphasize the elimination of label-label and label-symbol conflicts, while posting as many labels as possible. Therefore, the costs of label-label and label-symbol conflicts are both higher than the cost of deleting a label from the map, and the cost of deleting a label from the map is higher than the cost of moving it around its corresponding symbol.

Every possible position for a point feature label relative to its symbol has an associated cost. These costs are stored in a "position cost array" for each feature. Position costs are integers between 0 and N-1, where N is the number of possible label positions. So the first factor of the objective function encourages labels to stay close to their preferred position.

To handle label deletion, each label has an extra "position" which corresponds to deletion from the map. That position is assigned a cost two units higher than any cost corresponding to successful placement on the map. If the only way to resolve overplotting is to delete some labels, that happens as part of the normal execution of the annealing algorithm.

Label-symbol overplots are handled by changing the corresponding element in the label's position cost array to a value larger than the cost of deleting the label from the map. The randomized label position generation process never selects a label position with a cost higher than deletion. So any label is guaranteed to end up deleted from the map before it can overplot another feature's symbol.

Label-label conflicts are counted in the problem objective function as one unit higher than the cost of label deletion. In fact, any overplot is usually counted twice, because it is associated with two labels.

Changing the State of the System

The state of the label placement problem is changed by generating two pseudo random numbers. The first number selects a feature label to move. The second selects a new position for that label.

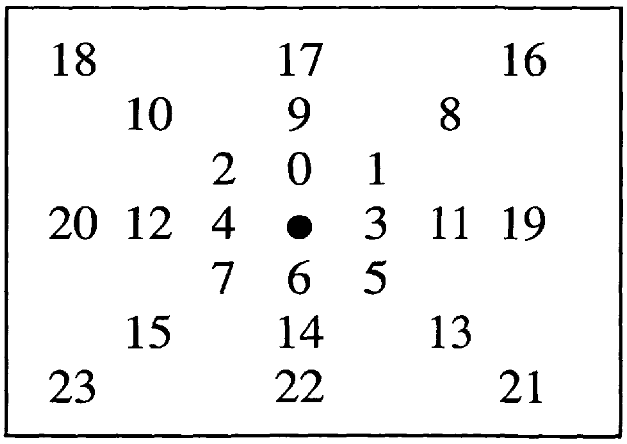

Figure 4. Twenty-four well label positions relative to a well symbol. This figure shows the general pattern. The program determines exact offsets as a function of label and symbol size.

Data Structures

The program was originally implemented in C and then modified to C++, using object-oriented technology where appropriate. I implemented a base Point-Symbol class and a Point-Feature class derived from the Point-Symbol class. Point-Symbol objects are just that: point symbols. Point-Feature objects are the combination of a cartographic symbol and a label.

Each Point-Symbol object has x and y coordinates, a radius, an identifying number, and a variable indicating how many possible positions are associated with the label for that symbol. The number of label positions is used in the program to differentiate between a Point-Symbol object and a Feature-Symbol object. The label position variable is set to 0 for Point-Symbol objects.

Each Point-Feature class object has the variables associated with a Point-Symbol object, plus a type variable for distinguishing between well, seismic, and seismic line name labels, a priority variable, a "rectangle" data structure with the height, width, and rotation angle of the label, a position variable, and a pointer to a dynamically allocated label position cost array.

Within the label placement program, all map features involved in the label placement problem are stored in a single, dynamically allocated "feature" array of pointers to Point-Symbol. As required, those pointers may be upcast to pointer to Point-Feature. Point-Symbols are distinguished from Point-Features by the fact that Point-Symbols have 0 label positions, while Point-Features have at least one label position.

Potential label-label conflicts are recorded in their own array indexed by feature and label position. For each label position for each point feature, all potential overplot conflicts with other labels are stored as pairs of integers. The first integer is the feature array index of another feature which has a label position which conflicts with the target label in its current position. The second integer is the conflicting position of the other feature's label. Many positions for a given feature's label may appear in the overplot array for another label. Information about each potential label-label overplot is stored twice, once for each feature. Given a feature with its label in a particular position, it is straight forward to search through the list of feature-position pairs which might cause conflicts. If a potentially conflicting feature label is currently in the conflicting position, a conflict does exist. Obviously in a production oriented program it is impossible to predict how many conflicts will be detected on a given map. Extra memory for recording label-label conflicts is allocated in blocks of 1000 words at a time.

The Annealing Schedule

Given an initial value for the temperature, T, the most commonly used annealing schedule is to compute new values of T from the old values by:

$$T_{new} = \lambda \bullet T_{old},\ 0 < \lambda < 1 \qquad (2)$$

More sophisticated schedules are described in academic publications. But they are not used much in real world applications—probably because simulated annealing is such a robust optimization tool. If it is going to work, identifying the perfect annealing schedule is usually not critical. (Furthermore, I suspect other users of simulated annealing algorithms spent most software development time as I did, writing code for the rest of the program, and were happy to accept any annealing schedule which performed reasonably well.)

For my label placement program I, like Christensen et al. (1995) and Edmondson et al. (1997) adopted the simple schedule shown in (2). Assuming the process does not terminate earlier due to some other criteria, I allowed 50 reductions of temperature with $\lambda = 9/10$. For each value of T, the program makes a number of random perturbations of the solution which is proportional to the number of label placement positions for the most complex label that is part of the problem. Usually, this proportionality factor is set to 10. If four successive reductions of the system temperature do not produce a better solution, the annealing process is stopped without finishing the normal 50 reductions (note 2).

Tag Lines Plotting

Because the program allows well labels to be placed far away from their symbols, "tag lines" are automatically created pointing from each well symbol towards the center of its label. Tag lines frequently cross labels for other wells. Plotting tag lines in a light gray and the labels in black satisfies the need to correctly associate label with symbol, without seriously reducing map legibility. Plotting tag lines as very narrow, solid black lines often works as well.

Overplot Detection

Three types of overplot conflicts may occur:

- Between pairs of rectangles outlining labels for two different point features;
- Between label rectangles and circles representing symbols for other point features, and

- As described below, crossing between tag lines joining label centers to their corresponding symbol center.

Probably the most efficient way to detect potential overplots would be to use three distinct "sweep line" algorithms. In the first sweep the program would detect overplots between label rectangles and symbols. In the second, the program would detect overplots between pairs of label rectangles, with the possible label positions restricted to avoid label-symbol conflicts previously detected. In the third sweep, the program would detect tag line crossings, again with the possible label positions restricted based on previously detected conflicts.

In this implementation I opted for simplicity over efficiency. The program performs all conflict detection in one sweep algorithm. The first step is to sort all the features from left to right on the x-coordinate of their symbol center. Next the program performs two nested do-loops. The outer loop is over the point features from left to right. The inner loop is over all point features to the right of the feature selected in the first loop, until the separation between the two feature symbol centers is so large that labels associated with the two features cannot possibly overlap. Inside the inner loop, all positions for the label of the first feature (indexed by the outer loop) are checked for possible overplots against the symbol and all positions for the label of the feature pointed to by the second loop index. The amount of work performed for each feature pair is constrained by a hierarchy of tests. For example, the program never checks for overplots between labels to the left of the first feature and to the right of the second feature. Also, the program checks for label-symbol conflicts first. If a label-symbol conflict has been detected and handled, there is no need to test that label position further for label overplots.

For well labels it is necessary to consider the possibility that even though label positions for two wells do not overlap, certain combinations of label positions may cause ambiguity. This can happen if the tag line drawn from one well symbol to the center of its label crosses a line drawn from another well symbol to the center of its label. The overplot detection code checks for such crossings, and two label positions are recorded as being in conflict if a crossing occurs.

Multi-line Labels

Some well labels will include several lines of text. Therefore, a Point-Feature class object includes a variable counting the number of lines in its label. If a label is made up of multiple lines, the normal single label rectangle structure is turned into an array of rectangle structures. The bounding rectangle of all the labels is stored in the first position of that array, and the rectangles for individual lines of text in the following array positions. Label-label and label-symbol overplot detection is performed first on the basis of the bounding rectangle, and then, if a potential conflict is detected, on the basis of the individual lines.

Actually, posting the multi-line labels required some special thought. Because a label stack can be moved far away from its symbol, it may be difficult to make a rapid visual association between labels and their symbols even using tag lines. I adopted the convention that if a label is directly above or below its symbol, the lines in the labels are center justified; if the label stack is to the right of its symbol, the lines of text are left justified; and if the label stack is placed to the left of its symbol, the stack is right justified. These conventions simplify the label-symbol association process and overall map readability, even though right-justified text is generally discouraged in graphic design.

Label Clusters

On any given map, point features may be distributed so that the choice of label placement for one subset of features has no impact on label placement for the remaining features. The label placement problem can be solved faster if each label subset is processed independently. Graph theory provides an efficient tool for dividing map features into subsets which cannot interact with each other during label placement.

A graph is a collection of vertices and edges. The vertices usually represent objects, such as point features on a map. The edges of a graph are connections between two vertices. To represent the label-label conflict problem as a graph, two vertices (point features) are connected by an edge if and only if some label placement for one of those features conflicts with a label placement for the other feature. Because one of the first steps in this program is to identify all potential conflicts between feature labels, it is easy to generate the graph representing features potentially in conflict with each other.

A path between two vertices in a graph is a list of vertices in which successive vertices are connected by edges. If there is a path between every pair of vertices in a graph, that graph is connected; otherwise, the graph is made up of connected subgraphs, some of which may be single vertices. If the graph used to represent the label placement problem is made up of connected subgraphs, label overplotting between labels for the features in each subgraph can be solved independently. Luckily, a well studied problem in graph theory is identifying the connected components of a graph. The depth search algorithm suggested in Sedgewick (1992) is the method implemented in this program.

Feature Priority Classes

A typical map will have features of varying importance. On most road maps, larger cities, highways and geographic features are considered more important than smaller ones. On those maps, labels for less important features are moved aside or deleted to make way for more important feature labels. On petroleum industry basemaps, well labels are more important than seismic line name labels and seismic shot point labels, and seismic line name labels are more important than seismic shot point labels.

Each point feature on the map has a priority code. Potential conflicts between all labels are recorded independently of feature priorities, but priority information is honored in evaluating the program objective function. During the evaluation of a change (Δ) in that function, conflicts between a high-priority feature and a low-priority feature do not count as part of the cost attributable to the high-priority feature. However, they do count as part of the cost attributable to the low-priority feature. As a result, low-priority feature labels are encouraged to move out of the way of high-priority feature labels, while placement of the high-priority feature labels proceeds independently of lower-priority feature labels.

Map Design Conflicts Between Labels

As mentioned earlier, petroleum industry mapping standards allow for seismic line name labels to be posted at both ends of a seismic line or at only one. For those companies where both labels are not wanted, a "pseudo" overplot conflict is defined to exist if both these labels are on the map. This is done by looping over all positions for the first label, and marking them as being in conflict with all positions for the second label.

An Initial Value for T

On the first 50 passes through the annealing schedule, the algorithm accepts any non-improving move with probability 1/3. At the same time the sum of the non-improving Δ values encountered during the 50 passes is recorded and used to compute an average value for non-improving Δ. The program then solves (1) for T so that p is 1/3 for the average value of Δ:

$$T = \bar{\Delta}/(\ln 3) \qquad (3)$$

From then on, T is reduced according to the schedule given above.

The Label Reduction Factor

A small amount of overplotting between the "edges" of labels does not seriously affect overall map legibility. Usually it is reasonable to trade some overplotting for more labels successfully placed on the map. The program supports a "label reduction" factor calculation to facilitate this option. Inside the program the dimensions of any label are reduced by the specified factor which is set to 5% by default. Label-label overplots and label-symbol conflicts are searched based on these reduced label rectangles. Small amounts of overplotting are not detected.

Program Performance

The program described in this paper has been tested against a wide range of randomly generated PFLPs, and against many real petroleum industry data sets. Several problems involved more than 1,000,000 potential label-label conflicts. Examples of performance against randomly generated problems were shown earlier. This section presents results using real data.

Figure 5 shows 44 clustered well symbols with 18 label positions allowed for each label. The larger map from which this cluster was drawn had 763 wells divided into nine independent clusters and over 10,000 potential label-label overplots. Overplot resolution was accomplished in less than 12 CPU seconds. Only one label was deleted from this particular cluster.

Figure 6 shows a detail from a seismic survey map with 2,023 shot point labels and 63 line name labels. Total CPU time for overplot resolution was 8 CPU seconds and 261 labels were deleted from the final map.

Figure 7 shows a detail from a large map with 2,448 wells. Based on conflicts between well labels, they are grouped into 70 clusters. Eighteen positions were allowed for each well label, and a total of 980,000 potential label-label conflicts were identified. Total solution time for the entire map was 7

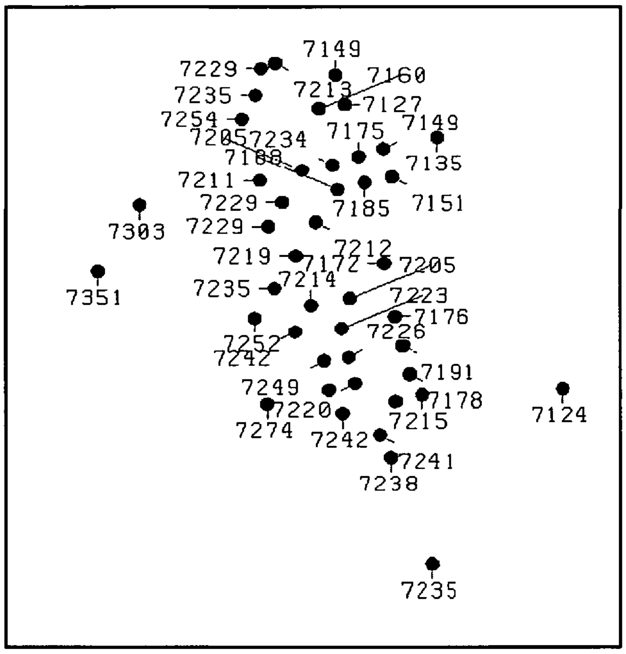

Figure 5. Forty-four clustered well symbols with associated labels. Eighteen label positions are allowed for each label. This is one cluster out of 9 on a larger map which had 763 wells, and over 10,000 potential label-label conflicts. Total CPU time to detect and resolve label overplots for the entire map was less than 12 seconds. For this example, the optimal label position was below its symbol.

CPU minutes. An interesting characteristic of this map is that at map scale, many of the well symbols essentially lie on top of each other. The program still successfully placed the majority of the labels on the map (708 were deleted), even if some labels were placed far from their symbol.

Figure 8 shows a detail from a map where each well feature label is made up of three lines of text. The larger map had 146 point features and label over-plot resolution—allowing 18 positions for each three-line label—took less than 10 CPU seconds.

The program spends significant CPU time in just two places: overplot detection and optimization of label placement. There is approximately a 50-50 breakdown between those two tasks. Other parts of the program, such as gathering information from the input map, generating the output map, or computing the connected components of a problem graph never require more than 20% of the total time.

Naturally, I was able to compare the performance of this algorithm against the integer programming algorithm by Zoraster (1990) based on Lagrangian relaxation and subgradient optimization. Differences were generally as clear-cut as implied by Christensen et al. (1995). The simulated annealing program ran faster and deleted fewer labels in every test. For example, on the seismic data set shown in Figure 6, the older algorithm required 12 CPU seconds and deleted 320 labels—as opposed to 8 seconds and 261 labels for the new program. On the clustered data set shown in Figure 5 (allowing only 8 positions for labels on the map, the maximum allowed in the older program), the simulated annealing program executed in 1.25 CPU seconds and deleted 10 labels, while the integer programming program used 4 CPU seconds and deleted 13 labels.

However, the most important difference between the simulated annealing and the integer programming approach are features of the C/C++ language, not speed or number of labels deleted. The older program is implemented in FORTRAN 77 which does not support dynamic memory allocation. Too often, users of that program discovered that not enough memory had been allocated to store information about label-label or label-symbol conflicts for very large problems—even though only eight positions were allowed for any single well label. Since this was not a problem in the simulated annealing

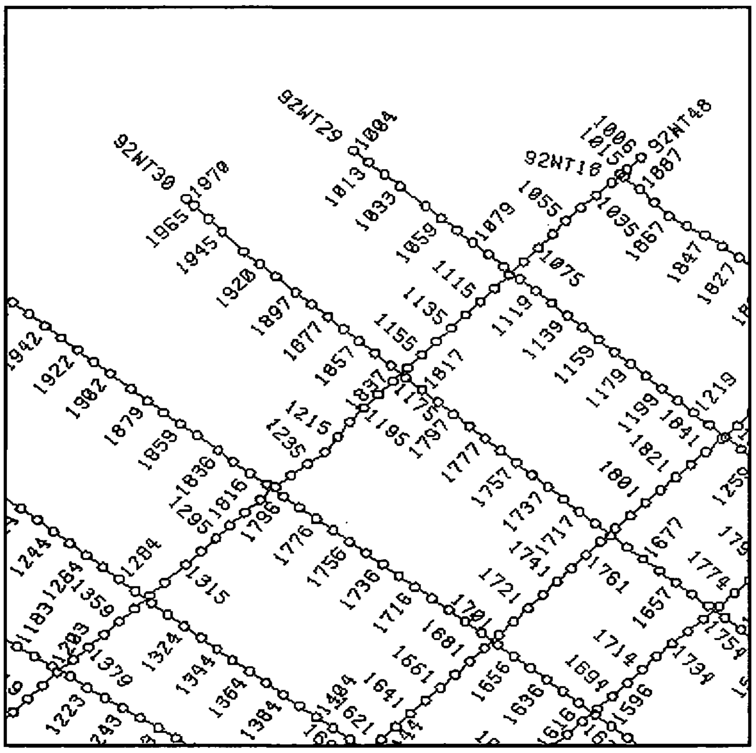

Figure 6. Seismic lines with shot point labels and line name labels after overplot resolution.

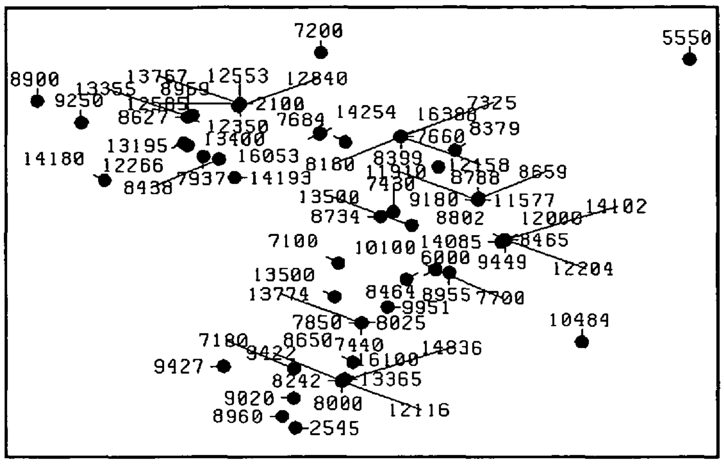

Figure 7. Detail showing application of program to a map with 2448 point feature labels and more than 900,000 potential label-label overplots. On this map several point feature symbols lie almost on top of each other, so that some features appear to have multiple labels.

implementation in C++ with dynamic memory allocation, that program can allow more label positions (up to 24) around each well symbol, making it much more useful for densely clustered maps.

Discussion

The key to the success of this program, as with any program which solves an optimization problem using simulated annealing, is implementing a problem objective function which can incorporate problem constraints and can be updated efficiently. Christensen et al. (1995) had already done most of the work, and I was able to build some extra functionality on top of their basic objective function.

The annealing part of the program was easy to implement once the objective function was identified. And once the annealing subroutines were working correctly, it was easy to add functionality, such as different types of labels and label type hierarchies, with minimal impact on the annealing code. However, the need for efficient updating means that all potential overplots between all point feature labels must be determined at the start of the program. Writing code to detect all potential label-label and label-symbol conflicts, and making that information easily accessible, was the hardest part of the implementation.

Is this program as efficient as it could be? Almost certainly not. Improvements are possible in that part of the program which finds the label-label and label-symbol overplots, and in the annealing schedule itself. However, the two tasks take up about equal amounts of CPU, and between them account for the overwhelming majority of time in the program execution. To achieve significant speed improvements overall would require making major improvements to the efficiency of both tasks.

Finally, although the maps to which this program were applied were all petroleum industry basemaps, nothing in the program design prevents it from being used for other types of maps. The overplot detection and simulated annealing algorithm would work equally well on other types of maps—and other types of map features—as long as a reasonable set of label positions can be generated and ranked for each feature label. This assertion is supported by the results obtained by Edmondson et al. (1997) in applying similar techniques to line and area map features.

Conclusions

Simulated annealing provides an efficient and robust method for solving PFLP problems encountered in mapping for the petroleum industry—even for maps involving thousands of labels, many possible positions for each label, hundreds of thousands of potential label overplots, clustering of point features, and other real world problems. Label placement programs

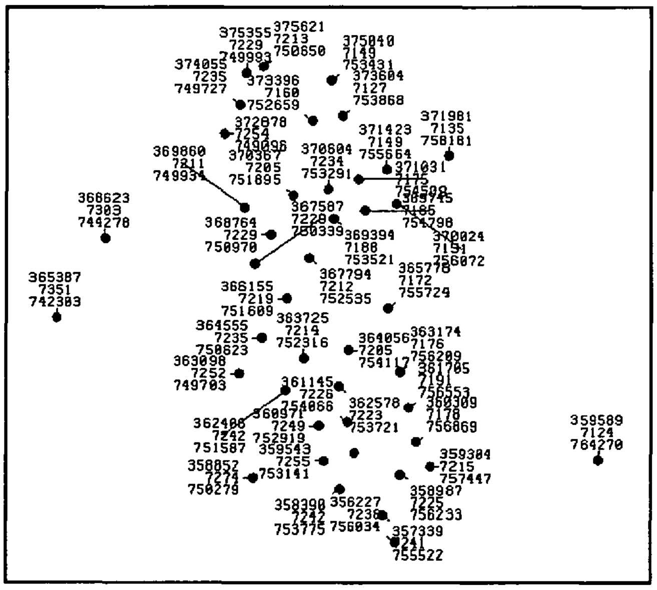

Figure 8. Overplot resolution involving stacked labels. Note that overplot detection is done on the level of the individual lines of text, so that a point feature symbol may appear to be surrounded by text from another feature's label. Also, for this problem, tag lines were allowed to cross.

using simulated annealing are easy to implement, with the hardest programming work involved in detecting potential label-label and label-symbol overplots.

NOTES

Note 1

All examples in this paper were created on a Silicon Graphics/Extreme workstation. The program has also been used on workstations made by SUN and IBM.

Note 2

Holding T fixed for a number of iterations and then reducing it by a constant factor is a "quenching" schedule rather than an annealing schedule. See Ingber (1993) for a discussion of the differences between annealing and quenching.

REFERENCES

Ahn, J. K. 1984. *Automatic map name placement system.* Ph. D. Dissertation, Rensselaer Polytechnic Institute, available through University Microfilms International, Ann Arbor, Michigan.

Ahn, J., and H. Freeman. 1984. A program for automatic name placement. *Cartographica* 21(2/3): 101-9.

Barnhart, C. L. 1962. *The American college dictionary.* Random House: New York.

Christensen, J., J. Marks, and S. Shieber. 1995. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics* 14(3): 203-32.

Cook, A. C., and C. B. Jones. 1990. A prolog rule-based system for cartographic name placement. *Computer Graphics Forum* 9: 109-26.

Cromley, R. G. 1985. An LP relaxation procedure for annotating point features using interactive graphics. *Proceedings of Auto-Carto* 7, Washington D. C. pp. 127-32.

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

Edmondson, S., J. Christensen, J. Marks, and S. Shieber. 1997. A general cartographic labeling algorithm. *Cartographica* (in press).

Formann, M., and F. Wager. 1991. A packing problem with applications to lettering of maps. *Proceedings of the Seventh Annual Symposium on Computational Geometry,* North Conway, N. H., July 1991. pp. 281-8.

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

Imhof, E. 1975. Positioning names on maps. *American Cartographer* 2(2): 128-44.

Ingber, A. L. 1993. Simulated annealing: Practice versus theory. *J. Math. Comput. Modelling* 18(11): 29-57.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science* 220(4598): 671-80.

Marks, J., and S. Shieber. 1991. The computational complexity of cartographic label placement. 1TR-05-9, Center for Research in Computing Technology, Harvard University.

Mower. 1989. *The selection, implementation, and evaluation of heuristics for automated name placement.* Ph. D. dissertation, The State University of New York at Buffalo, available through University Microfilms International, Ann Arbor, Michigan.

Nemhauser, G., and L. Wolsey. 1988. *Integer and combinatorial optimization. New* York: John Wiley & Sons.

Sedgewick, R. 1992. *Algorithms in C++.* New York: Addison-Wesley Publishing Company.

Siarry, P., B. Ludovic, and G. Dreyfus. 1987. Thermodynamic optimization of block placement. *IEEE Transactions on Computer-Aided Design* 6(2, March 1987).

van Roessel, J. W. 1989. An algorithm for locating candidate labeling boxes within a polygon. *The American Cartographer* 16(3): 201-9.

Zoraster, S. 1990. The solution of large 0-1 Integer programming problems encountered in automated cartography. *Operations Research* 38(5): 752-9.

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