Geography, University of Washington, Seattle.
Stenhouse, H., 1979. "Selection of Towns on Derived Maps." *The Cartographic Journal*, vol 16, June 1979: 30-39.
Toepfer, F. and W. Pillewitzer, 1966. "The Principles of selection." *The Cartographic Journal*, vol 3, June 1966: 10-16.
Vasmut, A.S., 1967. "Modeling the Process of Producing Topographic Maps." *Geodesy and Aerophotography*, 1967: 222-227.

NAME PLACEMENT OF POINT FEATURES THROUGH CONSTRAINT PROPAGATION

James E. Mower
University at Buffalo
Buffalo, New York 14214
U. S. A.

Abstract

The prospect of creating a fully automated map production system for the user with little or no experience in cartographic design presumes the development of a fully automated routine for performing name placement. This paper describes an algorithm for the propagation of name placements for point features which resolves space competition conflicts through labeling adjustments based upon dependency related backtracking or through the selective deletion of features in densely clustered areas.

Introduction

Our exposure to the ever increasing quality and availability of computer output devices has stimulated us to develop mapping software that can make use of these new capabilities. As the number of computer generated cartographic products has increased, so have efforts to produce them under standards that cartographers have long applied to manual production techniques.

Several of the major commercial statistical packages already offer graphic output, in the form of charts or largely user-created maps. We can increase the usefulness of these graphics procedures by minimizing user interaction, assuming that the users are largely uninterested in the details of map design. Other applications that will require fully automated map production are systems for performing automatic map-based navigation as well as for mapping rapidly changing phenomena.

Many aspects of map production are well suited to the application of algorithms, that is, procedures that are known to unambiguously and effectively solve problems and then terminate. Examples of such activities include contouring and hill shading. Algorithms to perform these activities simulate an objective set of discrete procedures or steps that cartographers could also use to solve the problems manually.

Unfortunately, there are several important cartographic activities for which objective procedures are lacking. One such activity is name placement. Although guidelines for good name placement exist, they are often stated and understood within an intuitive or subjective framework (Imhof, 1975; Yoeli, 1972). And, more to the point for algorithm development, few guidelines suggest a stepwise ordering of name placement tasks. Like Yoeli and Freeman and Ahn, we may propose a guideline that names be ordered in terms of the sizes and functions associated with them (Yoeli, 1972; Freeman and Ahn, 1984). We would then place areal ames before point names and larger names before smaller names. 'he application of such guidelines or heuristics does not uarantee that our algorithm will move forward without making a ıistake in a placement. Rather, they characterize a best guess r a rule of thumb for solving a problem.

A heuristically driven program must be able to recover from rrors that occur when a best guess does not provide the best olution at some stage of execution. Typically, such programs eturn (backtrack) to a state before the problem occurred and hen move forward with a different approach. To backtrack fficiently, a program must have the capability to record the hoices that led to the current problem (Winston, 1984). Several ecent approaches to name placement have used backtracking echniques (Freeman and Ahn, 1984; Ahn , 1984; Lewis , 1982).

We can limit the amount of backtracking within a procedure y taking advantage of natural constraints that are associated 'ith a problem. Because cartographers work with a finite map pace, the addition of name A to that space necessarily onstrains the latter positioning of other names within the eighborhood of the feature that A represents. The neighborhood f a feature refers to the polygon delimiting all possible ılacements of the name of the feature. If a placement conflict ccurs, the resolution of the conflict will be limited to the eprocessing of the neighbors of the changed feature.

Considerations for Placing Point Feature Names

Several published guidelines for the placement of names for oint features rank the quality of a placement in terms of its osition relative to the feature it represents. Imhof, for xample, suggests that the best placement for a label is to the pper right of the point symbol. He further suggests that artographers avoid placing names off to the left of a feature. 'oeli agrees that the best placement is to the upper right of a eature, but ranks some placements to the left of a feature as ıore desirable than others to the right. We can similarly iscriminate between good and bad placements on their distances rom the features they represent. Imhof notes that the distance f a name from its feature varies on the graphic nature of the ıap, map scale and the sizes of the objects and the typefaces by 'hich they are named. We must also require that a name be nambiguously associated with its feature. Of course, we must revent names from overlapping other names, features or symbology f the same color. Because the current paper will be limited to discussion of constraint propagation methods for generating ame placements, we will consider only the cases in which ılacements interfere with other placements or point features.

Using Constraints to Analyze Line Drawings

Like people, computers can make use of natural constraints to resolve ambiguous situations. When we move our head and notice an object which appears to shift against another "stationary" object, we infer that the shifting object is closer to us than the stationary object. Likewise, we are more apt to understand a diagram representing three dimensional figures if those figures are accompanied with representations of shadows.

Waltz has shown that a computer can generate descriptions of objects from line drawings given a set of constraints (Waltz, 1975). These constraints operate in two different senses. First, we impose a set of restrictions upon the images that the computer will process. We do not permit the occurrence of view specific junctions, that is, junctions between objects that change radically with a slight change in the position of the viewpoint. We also limit the number of possible line junction types to a small, finite number. Waltz's first procedure made use of only 18 such junctions. Second, the labeling of a line junction is constrained by the labels of its neighboring junctions. A line extending from a junction is compatible only with a line of the same type extending from a neighboring junction. Lines may represent concave edges, convex edges, or boundaries between objects and the background.

Waltz's procedure begins by visiting a junction and assigning it all possible geometrically correct labels. It then finds the neighbors of the current feature and puts them on the processing stack. As a junction is popped off the stack for inspection, the procedure checks whether it has previously inspected it. If not, it assigns the junction a set of possible labels and then puts its neighbors on the stack. If it has visited the junction, it checks whether any of its labels conflict with all possible labels at any neighboring junction, deleting any offending labels from the current junction. If a conflict occurs, it puts all the neighbors of the current junction on the stack. The procedure stops when no junction has labels which conflict with all of the labels of any of the neighboring junctions.

Waltz's procedure uses local conflict resolution to prevent the reprocessing of junctions that are not involved in conflicts. We argue that cartographers use similar strategies for resolving name placement conflicts on maps. When a cartographer encounters insufficient space for labeling a map feature, he or she will adjust the labels of other features in the local area until all of the labels, including the current one, are in suitable locations. Since the cartographer may wish to reserve placements for important or difficult to place features, it is likely that he or she will adjust the labels of other local features with fewer constraints. If the cartographer cannot generate sufficient space to place the current feature through adjustment, he or she must delete the current feature or one or more of its neighbors. The cartographer will likely delete the feature or features of the least importance for the purpose of the map.

The following algorithm makes use of local conflict resolution, measures of feature importance and local labeling constraints to automatically place point features and their labels on maps.

An Algorithm for Placing Point Names

To find the proper name placements for point features:

```
1 Form a stack of all point features, ordered by their degrees
  of freedom.
2 Until the stack is empty:
   2.1 Remove the top element from the stack. Call it the
       current feature.
       2.1.1 If the current feature has never been visited,
             assign it the best placement under any arbitrary
             ranking system. Note a change has occurred.
             2.1.1.1 While the current placement of the
                     current feature overlaps another point
                     feature, move the current placement to
                     the next best placement.
                     Record interfering features.
             2.1.1.2 If no position can be found to place the
                     name:
                     2.1.1.2.1 For every placement with no
                               features of greater
                               importance than the current
                               feature, count the number of
                               interfering features at each
                               placement.
                     2.1.1.2.2 If every placement contains a
                               feature of equal or greater
                               importance than the current
                               feature, delete the current
                               feature. Note a change has
                               occurred.
                     2.1.1.2.3 Otherwise, delete all features
                               at the interfering placement
                               and place the current feature
                               there. Note a change has
                               occurred.
       2.1.2 Otherwise, if the current placement of the
             current feature overlaps the current placement of
             a neighboring feature, update the current
             placement for the feature of lesser importance.
             Note a change has occurred. If no more
             placements exist for the feature of lesser
             importance, delete it. Note a change has
             occurred.
   2.2 If a change has occurred, put the neighbors of the
       changed feature on top of the stack.
3 For every feature, label the feature by inspecting its
  current placement.
```

We begin by reading a file containing information about the point features. For each feature we read its name, location, point size of its label and a measure of its importance. For the current example we will use population figures to assign relative importance to point features. We then compute the area of the polygon surrounding the feature such that all possible label placements for the feature fall within the polygon. Following Freeman and Ahn, we then sort the record by the area of its polygon (Freeman and Ahn, 1984). We repeat this process until we come to the end of the file.

Having read and sorted the feature information, we hold it for processing in a stack data structure. Conceptually, a stack is a list of data open at only one end for processing. We can both add information to the stack and remove it for inspection. The last element of information that we "push" on the end of the stack will be the first element that we "pop" off for inspection.

We begin the placement procedure by popping the endmost feature from the stack. Calling it the current feature, we first determine whether we have visited the feature previously as a current feature. If not, we assign it the best placement possible under whatever criteria for placement we choose and note that a change has occurred. The present criteria is a slight modification of one that Yoeli has proposed, consisting of eight possible label positions with associated rankings of goodness (see figure 1).

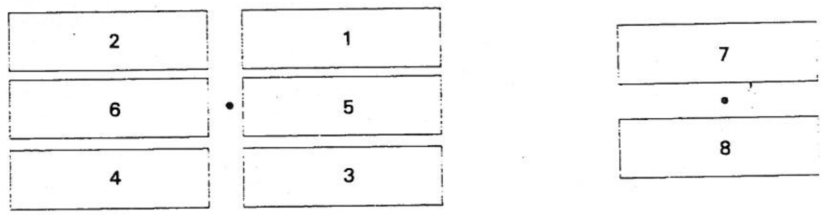


Figure 1. Ranking of point feature placements (from Yoeli, 1972)

Having started by placing the label at its best location, we check whether that placement will overlay any of the point symbols for features that are on the stack. Without removing the features from the stack, that is, without disturbing the position of the stack pointer, we check whether the placement polygon of the current feature intersects the polygon of any of the other features on the stack. For any intersecting feature, we record its name and check whether the current placement will overlay the point symbol of the conflicting feature. If so, then we move the placement of the current feature to the next best placement. We continue checking for point symbol overlaps until we find a placement with no conflicts or until we eliminate the last placement. Upon eliminating the last placement, we try again, beginning at the best placement, by cycling through the placements, counting the number of overlaps at each location and checking the importance of the features that interfere with the current feature at each placement. We then identify the placement having the least number of conflicts and containing no feature of greater or equal importance than the current feature. We delete those features within that placement and substitute them with the label for the current feature. If we can not fi placement for the current feature that meets both of these :riteria, then we delete the current feature. Because the ılgorithm selectively deletes features that it cannot place, we lo not need to store multiple, scale dependent data bases.

If we have visited the current feature previously, we check o see if its current placement interferes with one or more of he placements of its neighbors. If so, we resolve the conflict y moving the placement of the feature of lesser importance, gain checking whether this new placement interferes with eighboring point symbols. If the feature to be moved is already abeled at its worst placement, then delete that feature. uture versions of the algorithm may update the more important eature if the less important feature is at the worst placement. t may also reduce the point size of a feature if the reduction ould not interfere with the analogical function of the ettering. If we have moved a placement or deleted a feature hen we note that a change has occurred.

Because the resolution of a labeling conflict may create ther labeling conflicts within the neighborhood of the changed eature, we must sort the neighbors of the changed feature and ush them back onto the stack if they are not there already. ere we encounter elements of both backtracking and local rocessing. We often find that map space constraints prevent us rom implementing our heuristic of placing every feature at its est location. We must occasionally backtrack and move lacements to fit other features. By only considering features n the neighborhood of the changed feature, we avoid reprocessing eatures that have nothing to do with the current conflict. We ould not expect a cartographer to reposition labels in areas of he map representing sparsely populated places to resolve abeling conflicts in densely populated areas.

Current Implementation and Results

An implementation of the algorithm in the C programming anguage is currently operating on a Digital Equipment VAX 780 inicomputer, a Sperry 7000/40 minicomputer and an IBM-AT icrocomputer at the University at Buffalo. Both the VAX and perry computers are running the Berkeley UNIX 4.3 operating ystem. The IBM-AT is running the Microsoft XENIX operating ystem. The implementations for each of the three computers are lentical.

We used a Compugraphics MCS Power View 10 computer pesetting system to generate the accompanying diagrams. We nerate character strings within the name placement program that rrespond to operating system commands on the typesetter. We ıen download the program output with the typesetter commands and acement information to a floppy disk. The floppy disk is then ansferred manually to an IBM-XT microcomputer which serves as a ta link to the typesetter. After running a translation program ı the typesetter to remove carriage returns and other interpretable control characters from the file, we phototypeset

the information. The program output includes the name of the place, its location, the location of its label and its point size.

The input data currently consists of 44 towns from Monroe County in New York. We digitized the locations of the towns from a road map on a scale of 0 to 10,000 digitizer units on both the X and Y axes. By using 5 significant digits to the left of the decimal, we avoid performing floating point multiplication or division upon the data. After it has performed all of the computations, the program translates the output coordinates to picas and points for the typesetter.

We are currently planning to acquire a digital gazetteer of the populated places of New York. Until we are able to use a data set of this magnitude as input, we are largely unable to comment on program performance factors. However, we note that running times, though notoriously unreliable on time sharing systems, are relatively constant over a wide range of scales. Therefore, we expect that running times using larger data sets will rise linearly with the number of features. Running times on all 3 computers average under 3 seconds of real time for processing the 44 name data set.

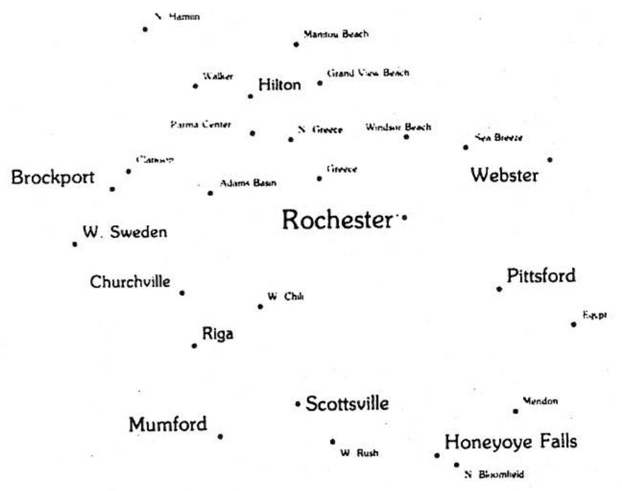
Directions for Further Research

This algorithm for the placement of point features demonstrates the feasibility of using constraints, local processing, and backtracking techniques to efficiently place point features. We expect to successfully apply the same techniques to place line and area features as well. Like Freeman and Ahn, we intend to place area features before point features, given that constraints on the placement of area features are generally more severe than those on point features. We will place linear features last.

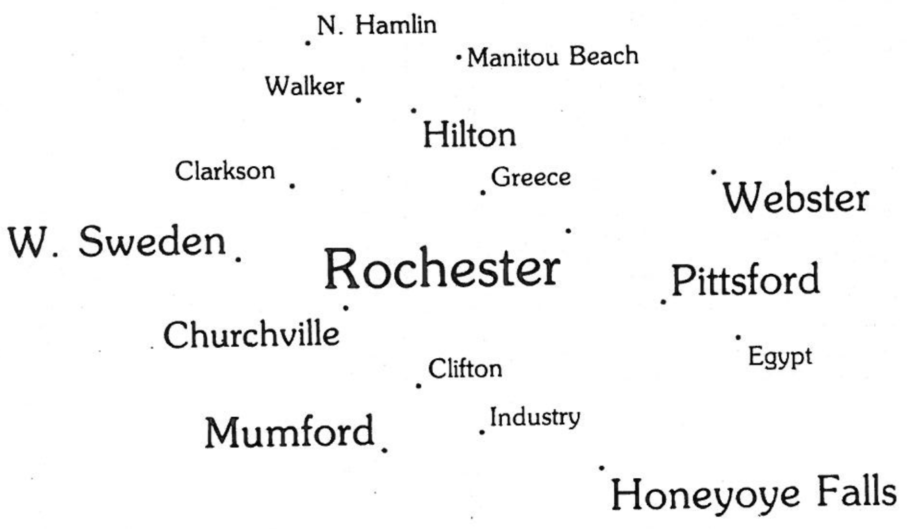
Because the algorithm will place point features after area features, it must make the positions of the area names available to the point placement procedure. Rather than treating the area names as contiguous wholes, the algorithm will record the positions of the letters of the area names as rectangles of fixed positions. Whenever the placement of a point feature conflicts with the placement of an area name, the point feature label will always be moved, regardless of its importance. We will likewise make the positions of area and point names to the line placement procedure.

Acknowledgements

David Mark of the Department of Geography at the University at Buffalo offered many helpful comments during the development of the algorithm and its implementation. Greg Theisen of the University at Buffalo Cartographic Laboratory provided invaluable assistance with the typesetter-microcomputer interfaces.



igure 2. Placement of point features for Monroe County, New York. Minimum point size of 6 and maximum point size of 16 before reduction.



igure 3. Minimum point size of 22 and maximum point size of 40 points before reduction.

References

Ahn, J., 1984. "Automatic Name Placement System." Doctoral Dissertation, Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, New York.

Cromley, Robert G., 1985. "An LP Relaxation Procedure for Annotating Point Features Using Interactive Graphics." Proceedings, Auto-Carto 7. Falls Church, Virginia: American Society of Photogrammetry and American Congress on Surveying and Mapping.

Freeman, H. and J. Ahn, 1984. "Autonap--An Expert System for Automatic Name Placement." Proceedings, First International Symposium on Spatial Data Handling. Zurich: Geographical Institute, University of Zurich.

Hirsch, S. A., 1980. "Algorithms for Automatic Name Placement of Point Data." Master's Thesis, Department of Geography, State University of New York at Buffalo, Buffalo, New York.

Imhof, E., 1975. "Positioning Names on Maps." The American Cartographer, vol. 2, no. 2, pp. 128-144.

Kelly, P. C., 1980. "Automated Positioning of Feature Names on Maps." Master's Thesis, Department of Geography, State University of New York at Buffalo, Buffalo, New York.

Lewis, G. E., 1982. "Automated Point Labeling for Geographic Data Bases." Master's Thesis, Department of Geography, Western Washington University, Bellingham, Washington.

Pfefferkorn, C., D. Burr, D. Harrison, B. Heckman, C. Oresky and J. Rothermel, 1985. "ACES: A cartographic Expert System." Proceedings, Auto-Carto 7. Falls Church, Virginia: American Society of Photogrammetry and American Congress on Surveying and Mapping.

Waltz, D., 1975. "Understanding Line Drawings of Scenes with Shadows," The Psychology of Computer Vision, ed. P. H. Winston. New York: McGraw Hill Book Company.

Winston, P. H., 1984. Artificial Intelligence. Reading, Massachusetts: Addison Wesley Publishing Company.

Yoeli, P., 1972. "The Logic of Automated Map Placement." The Cartographic Journal, vol. 9, no. 2, pp. 99-108.