Algorithms for three-label point labeling
by Robert Wade Duncan

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Montana State University

Abstract:
Automated map labeling has become an area of increasing interest in computational geometry. This is largely due to recent advancements in interactive graphical visualizations and Geographical Information Systems (GIS). This thesis studies the problem of labeling a map so that each point feature associated with the map receives three square-labels of maximum size. The motivation is that in some applications, such as a weather map, each point must contain three pieces of information. This thesis presents algorithms that yield solutions for both the discrete model (each point lies at a comer of its square labels) and the sliding model (each point lies on the boundary of its square labels). For the discrete model the algorithm meets the lower time bound of Ω(n log n) and for the sliding model we present an O(n3 log n) time algorithm. We also discuss implementation of the algorithms presented.

ALGORITHMS FOR THREE-LABEL POINT LABELING

by

Robert Wade Duncan

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

May 2001

APPROVAL

Of a thesis submitted by

Robert W. Duncan

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliography style, and consistency, and is ready for submission to the College of Graduate Studies.

Binhai Zhu, Ph. D. _____ (Signature) May 3, 2001 Date

Approved for the Department of Computer Science

J. Denbigh Starkey, Ph. D. _____ (Signature) May 3rd 2001 Date

Approved for the College of Graduate Studies

Bruce McLeod, Ph. D. _____ (Signature) 5-4-01 Date

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Signature Robert W. Dunn

Date 05/04/01

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Automated map labeling has become an area of increasing interest in computational geometry. This is largely due to recent advancements in interactive graphical visualizations and Geographical Information Systems (GIS). This thesis studies the problem of labeling a map so that each point feature associated with the map receives three square-labels of maximum size. The motivation is that in some applications, such as a weather map, each point must contain three pieces of information. This thesis presents algorithms that yield solutions for both the discrete model (each point lies at a corner of its square labels) and the sliding model (each point lies on the boundary of its square labels). For the discrete model the algorithm meets the lower time bound of $\Omega(n \log n)$ and for the sliding model we present an $O(n^3 \log n)$ time algorithm. We also discuss implementation of the algorithms presented.

CHAPTER 1

INRODUCTION

Map labeling has been an important area of research since the beginning of the art of cartography. In recent years the revised problem of automated map labeling has become the focus of many geometers. The ACM Computational Geometry Task Force has named it as one of the more important topics of research in Discrete Computational Geometry [B96]. Practical applications have arisen in multiple areas including interactive graphical visualizations and Geographical Information Systems (GIS).

Map Labeling Background

Map labeling is the task of placing labels at various features associated with a map. The objective is to determine label placements in order to maximize the quality of the visual appearance. It is generally accepted that the labeling problem can be divided into three distinct tasks [I75]: (i) labeling *area* features (such as a body of water or landmass); (ii) labeling *line* features (such as a river or road); and (iii) labeling *point* features (such as a city or mountain peak). Although these problems are very distinct, the combinatorial aspects are independent of the feature being labeled. We can therefore focus on a single feature to label without loss of generality as shown in [CMS95, KT98a, KSWW01]. The quality of labeling has been studied most notably by Imhof [I75] and Yoeli [Y72]. Based on the results of their research the following two rules have formed the standard criteria for what constitutes a good labeling for point features: (i) for every label it is intuitively clear which point feature is described; and (ii) no two labels overlap. Using these two

basic requirements one can formulate a problem to either maximize the number of points labeled or label each point with maximum size labels. Each of these problems in some simplest form is NP-hard [FW91, MS91]. In this thesis we focus on labeling point features with maximum size labels.

Before we continue, we develop definitions necessary for an understanding of optimization problems. An *approximation algorithm A* for a maximization problem Π is said to have a *performance guarantee* of ρ if for any instance I of Π, $\rho \geq \mathrm{OPT}(I)/A(I)$, where $\mathrm{OPT}(I)$ is the optimal solution and $A(I)$ is a valid solution. If $\rho = 1$ we say A provides an *optimal solution.* The algorithm A is often referred to as an approximation algorithm for Π with a factor of ρ. Similarly, for a minimization problem Π, an approximation algorithm A has a factor of $\rho \geq A(I)/\mathrm{OPT}(I)$. (For more information regarding optimization problems and approximation algorithms refer to [GJ79].)

Formann and Wagner studied the point feature labeling problem PFLP by showing that it is NP-complete in one of its simplest forms [FW91]. (The reader is referred to [GJ79] for information regarding NP-Completeness or NP-Hardness). The problem was defined as follows: Given a set of points, assign each point an axis-parallel square label. Each square label must be placed such that the point feature it labels lies at one of its four corners. All squares are of uniform size, and the edge length of the uniform squares is maximized (Figure 1.1(a)).

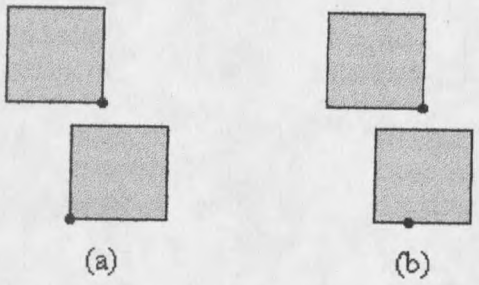


Figure 1.1 Example of the discrete and sliding models.

Formann and Wagner also showed that it is NP-hard to approximate the optimal solution within a factor of two. In other words, unless P = NP, no polynomial-time approximation algorithm can guarantee a solution within 50% of the optimal solution. Formann and Wagner also presented one such $O(n \log n)$ time algorithm [FW91]. Wagner next proved that it requires $\Omega(n \log n)$ time to reach this approximation bound [W94]. Wagner and Wolff then presented heuristic algorithms with favorable empirical results [WW95].

The point-labeling problem of Forman and Wagner has led the way for several variations. One such variation is a labeling problem where labels are placed so that their corresponding point lies on their boundary. This new model is in contrast to the previous model where labels are restricted to a position so that their corresponding point lies at their corners. This more natural model has an infinite number of square placement candidates, and adds to the complexity of the discrete model. Doddi et al. was the first to introduce the problem in its combinatorial form [DMMMZ97] and presented a polynomial-time approximation algorithm. The new model has been named the sliding model [KSW99] and has received much of the map labeling focus for the last few years [DMMMZ97, KSW99, ZQ00, DMM00].

Another generalization to the problem of map labeling has been based on orientation. Three distinct problems have arisen based on orientations [DMMMZ97]: (i) free labeling; (ii) oriented labeling; and (iii) fixed-oriented labeling. Free labeling places no extra restriction on the orientation of the labeling squares. Oriented labeling restricts the orientation of squares to have a common orientation. And fixed-orientation labeling requires that all labeling squares have a common and pre-defined orientation.

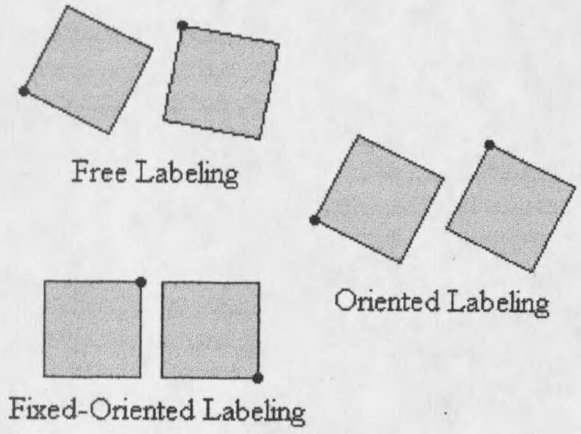


Figure 1.2 Example of the three orientation models.

Figure 1.2 shows a discrete labeling for each of the three orientation models. The first of the three problems, free-labeling model, was studied by Doddi et al. who found an approximation algorithm with a performance guarantee of about 36.6 [DMMMZ97]. The performance guarantee was first improved to 12.95 [ZQ00] and then to 5.09 [DMM00]. The second of the three problems, oriented-labeling model, was studied by Zhu and Qin who presented a factor-4 approximation algorithm [ZQ00].

Another variation to the map labeling problem was introduced by Kakoulis and Tollis who presented a heuristic algorithm for solving the problem of labeling point features with two labels [KT98b]. Poon and Zhu studied the problem labeling each point

with two axis-parallel squares and presented a factor-4 approximation algorithm, which runs in $O(n \log n)$ time [ZP99]. The performance guarantee was then improved to 3 [ZQ00] and then Qin, et al. further improved the results by presenting a factor-2 approximation algorithm [QWXZ00]. Qin, et al. also proved it is NP-hard to approximate the label size beyond a constant factor $\delta > 1$.

The problem of labeling points with two labels naturally leads to that of labeling point features with either three or four square-labels. The latter problem (labeling with four uniform squares) is trivial and not worthy of study. The problem of labeling point features with three squares, however, is interesting. It can easily be shown that for both a discrete and sliding model of this problem there is a lower bound of $\Omega(n \log n)$. We will now turn our attention to this problem. In the following section we give a definition for the problem of labeling a set of points with three axis-parallel squares.

Definition of the MLUST Problem

We define the problem to be studied in detail throughout the remainder of this thesis, Map Labeling With Uniform Square Triples (MLUST). Given a set of points in the plane, label each point with square triples (three uniform squares) of maximum size such that the following conditions are met.

(a) All squares are axis-parallel,

(b) All squares are of equal size, and

(c) No two squares intersect, except at their boundaries

This general problem naturally leads to two more specific problems. The first, MLUST under the discrete model, restricts placement of each of the three square labels so that their corresponding point feature lies at a corner of each square label. The second, MLUST under the sliding model, restricts the placement of the three squares such that their corresponding point feature lies on the boundary of each square label (not necessarily at a corner). We will discuss both problems in this thesis.

It is easy to see that the first problem has a finite number of placement candidates. In fact there are exactly four possible square triplet placements for each point feature. MLUST under the sliding model, however, has an infinite number of possible placements, and is the more complicated of the two. In the next section, we show that both variations of this problem have a lower bound of $\Omega(n \log n)$ time.

Lower Bound

It is an easy task to show that the MLUST problem under both the discrete and sliding models have a lower bound of $\Omega(n \log n)$. The following is a reduction from the element uniqueness problem, which has a lower bound of $\Omega(n \log n)$ under the algebraic computation tree model [SY82]. Given a set of real numbers $\{x_0, \ldots, x_n\}$ the set contains only unique integers if and only if the MLUST problem for the point set $\{(x_0, 0), \ldots, (x_n, 0)\}$ has a non-zero solution. In this thesis we present an optimal $O(n \log n)$ time algorithm for the discrete model and an $O(n^3 \log n)$ time algorithm for the sliding model.

Metrics and Definitions

In this section we specify the general metric to be used and define several terms that are needed for an understanding of the algorithms to follow.

Metrics

The L_∞-distance between two points is the largest of the differences between their coordinates. (For more information regarding L_∞-metric refer to [PS85]). Under our 2D model it represents $\max(|x_2 - x_1|, |y_2 - y_1|)$. For two points, p_i and p_j, the distance between the two points using the L_∞-metric is denoted $d_\infty(p_i, p_j)$ (Figure 1.3).

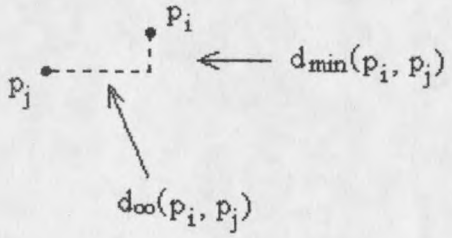


Figure 1.3 Example of $d_\infty(p_i, p_j)$ and $d_{min}(p_i, p_j)$.

Similarly the distance $d_{min}(p_i, p_j)$ is the smallest of the differences in the coordinates (Figure 1.3). Under our 2D model this represents $\min(|x_2 - x_1|, |y_2 - y_1|)$. For a set of points S, the distance $D_\infty(S)$ will represent the closest pair of points using the L_∞-metric. It is the minimum $d_\infty(p_i, p_j)$ of any two points in S. It should be noted that finding $D_\infty(S)$ can be accomplished in $O(n \log n)$ time using standard CLOSEST PAIR algorithms. One such algorithm is the divide-and-conquer technique detailed in Chapter 4.

Sub-squares and Labels

For the MLUST problem, each point $p_i \in S$ must be labeled with square triples. For the discrete model this implies that for each of the three square labels there are four possible candidates. We will call these four candidates *sub-squares* of p_i. To correctly label p_i we must select three of the four sub-squares as *labels* so that they do not intersect with the labels (selected sub-squares) of another point.

Pressure and Force

Whenever two points, p_i and p_j, have a pair of intersecting sub-squares, we say that the two points are exerting *pressure* on each other (Figure 1.4(a)). Point p_i can then label three of its sub-squares in such a way as to relieve any pressure it has on p_j (Figure 1.4(b)), or p_i can label three sub-squares so that it exerts *force* on p_j (Figure 1.4(c)). In this last case, we say that p_j accepts force from p_i if it can label its three remaining sub-squares.

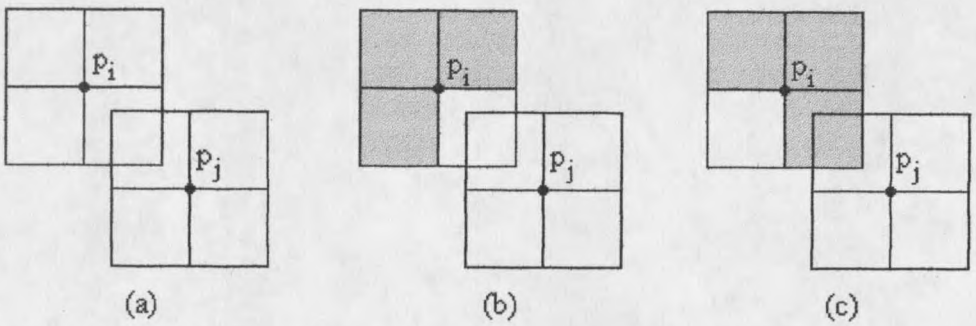


Figure 1.4: Example of pressure and force.

CHAPTER 2

MLUST UNDER THE DISCRETE MODEL

Our algorithm for solving this problem has three major steps. First, we show that the optimal edge length is bounded both above and below. This is necessary for obtaining linear-size graphs and keeping our solution within the $O(n \log n)$ time requirement. Second, we show that given a set of points and a fixed edge length, we can determine if an acceptable labeling exists in $O(n)$ time. And finally, we show that for any set of points there are linear candidates for the optimal edge length. A simple binary search for the largest acceptable edge length on the sorted list of candidates will result in an optimal $O(n \log n)$ time solution.

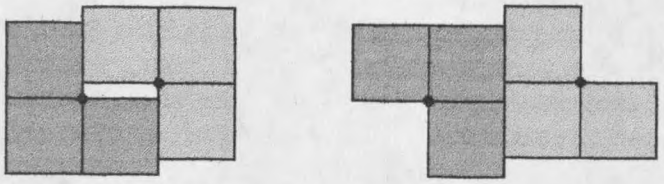
Linear Pairs of Concern

It is important for our algorithm when checking for sub-square intersections not to exhaustively check every pair of points. This would take at least $O(n^2)$ time. Instead, we intend to only check a linear number of point pairs. We first show the optimal edge length, l_{OPT}, is bounded both above and below. Next we show that due to this bound there are only a constant number of points that could intersect any given point. Thus, there are a linear number of point pairs that could intersect each other. We will refer to this list of point pairs as *pairs of concern*. This leads to our first lemma, which restricts the number of points that could intersect a given point to a constant number.

Bounding Optimal Edge Length

Lemma 2.1 $D_\infty(S)/2 \leq l_{OPT} \leq D_\infty(S)$

Proof. By the definition of $D_\infty(S)$ there must be two points, p_i and p_j in S, such that $d_\infty(p_i, p_j) = D_\infty(S)$. It is clear that if $l > D_\infty(S)$ there is no labeling for p_i and p_j that does not result in an intersection of labels, thus there is no acceptable labeling. Figure 2.1(a) shows a labeling of the closest two points in the set with $l = D_\infty(S)$. It is easy to see that l is optimal. Similarly, Figure 2.1(b) shows a labeling of two points where $l = D_\infty(S)/2$. It is clear that there is no labeling for the two points that could cause an intersection of sub-squares. Since these two points are the closest two points in the set, there can be no pair of points that have intersecting sub-squares. Thus, $l = D_\infty(S)/2$ must have a valid labeling and is the lower bound for l_{OPT}. ■



(a) $l = D_\infty(S)$ (b) $l = D_\infty(S)/2$

Figure 2.1 Example of the two points, p_i and p_j, where $d_\infty(p_i, p_j) = D_\infty(S)$.

Linear Number of Intersections

Lemma 2.2 *For a set S there are only 24n point pairs that could intersect each other.*

Proof. Using Lemma 2.1 it is easy to show that for a point, p_i, there are only a constant number of points whose sub-squares could intersect those of p_i. From Lemma 2.1 we know that two points, p_i and p_j, where $d_\infty(p_i, p_j) > 2 * l$ cannot have intersecting sub-

squares (refer to Figure 2.1(b) for an example). Also, remember that $l_{OPT} \leq D_\infty(S)$. Therefore, if $d_\infty(p_i, p_j) > 2 * D_\infty(S)$, the pair (p_i, p_j) cannot have intersecting sub-squares. This implies that if we draw a square, Q, centered at p_i with edge length $4 * D_\infty(S)$, then no point outside of Q could have an intersecting sub-square with p_i. Since all points must be at least $D_\infty(S)$ apart, we can pack at most twenty-four points within Q (Figure 2.2). If each point p_i can intersect with at most 24 other points, then there are at most $24n$ pairs that can have intersecting sub-squares. In Chapter 4 we present an algorithm for computing the $24n$ pairs of concern in $O(n \log n)$ time. ■

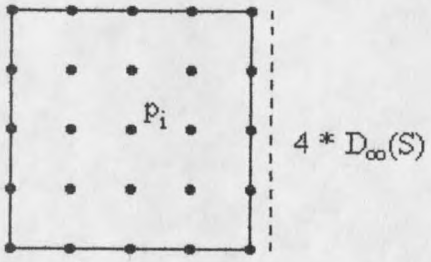


Figure 2.2 The maximum number of points within $2 * D_\infty(S)$ of p_i is twenty-four. Each pair of points in the figure is minimum $D_\infty(S)$ apart.

Determining If An Acceptable Labeling Exists

Our next goal, is to show that given a set of n points and an edge length, we can determine in $O(n)$ time if an acceptable labeling exists. Our strategy is to show that an acceptable labeling exists if and only if the intersection graph has no connected component that contains more than one cycle. We check for multiple cycles by creating a transformation graph with edges representing intersecting sub-squares and then use standard graph algorithms to check the transformation graph for multiple cycles.

Transformation

Given a set S of n points and a real value l we define the transformation graph $G(S, l)$ as follows. For each point in S, create a corresponding vertex in G. We then look at the sub-squares of edge length l for each point in S. Add an edge between two points in G whenever the corresponding points in S have a pair of intersecting sub-squares. If a pair in S has two intersecting sub-squares, add two edges between the corresponding vertices in G (Figure 2.3). If a point in S has more than two intersecting sub-squares, then stop, the set S has no acceptable labeling with edge length l.

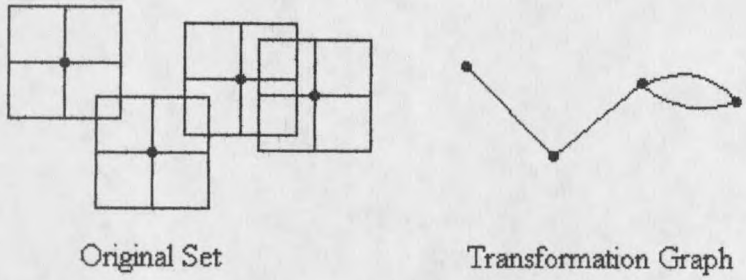


Figure 2.3 Example of a transformation graph.

To be a disconnected component of G implies that the corresponding points of S are not close enough to have any intersecting sub-squares, and thus can exert no pressure on each other. It is easy to see that the labeling of disconnected components have no influence on each other. We can therefore label each connected component separately.

Acceptable Force

A point can accept force in at most one of its sub-squares. A point may, however, have pressure in several sub-squares. If p_i only receives force in a single sub-square, thus destroying one of p_i's four sub-squares, it can still be labeled utilizing its three remaining

sub-squares. If the point p_i receives force (not simply pressure) in two of its sub-squares, either from a single point or two separate points, it can no longer be labeled. It should be noted that no two points may exert force on the same sub-square of p_i without either at least one of them also exerting force in a second sub-square of p_i (Figure 2.4(a)) or the two points having an illegal intersection of labels (Figure 2.4(b)). In either case no valid labeling exists. Therefore, p_i can be labeled if and only if it receives force from at most one point.

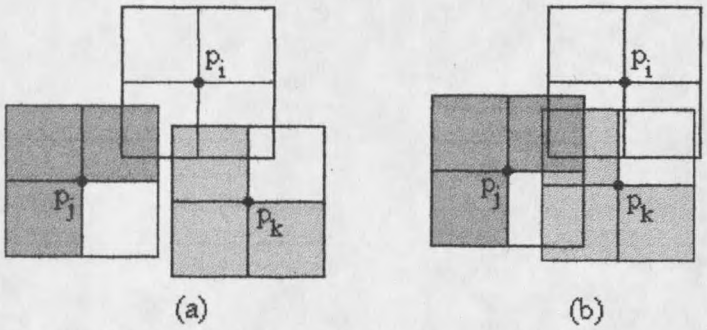


Figure 2.4 A point may receive force in at most one sub-square.

Lemma 2.3 *Any set of points S can be labeled with square triples of edge length l, if and only if no connected component of the transformation graph G has more than one cycle.*

Proof. We divide this proof into two parts. Part A will show that the set S can be labeled if its transformation graph G has no connected component with more than a single cycle. Part B will show that there is no labeling for S if G has a connected component with two or more cycles.

Part A: Any set S whose transformation graph G has no connected component with more than a single cycle can be labeled. Figure 2.5 shows a correctly labeled set of points with one cycle.

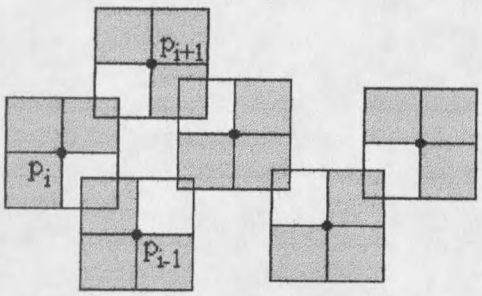


Figure 2.5 A component with at most one cycle has a valid labeling

In graph G, an edge represents pressure between two points in S. This implies that force may be exerted in either direction between the two points. It is easy to see that a connected component with no cycles can be easily labeled with the following procedure. Start at any leaf node, p_i, in the connected component. Exert force down every edge of p_i, and move to the points receiving force. At each of these new points you also exert force outward down every edge, except the edge you arrived on. Since there are no cycles, we will never visit a point twice, and thus never exert force on a point twice. Any connected component with exactly one cycle can be labeled with the following procedure. Choose a point in the cycle, namely p_i. Exert force from p_i to the next point in the cycle, p_{i+1}. At each new point continue to exert force following the cycle. When we arrive at the last point in the cycle, p_{i-1} we exert force from this point to our starting point, p_i. Each point in the cycle has now received pressure from exactly one point. If there are any other chains connected to the cycle, we simply exert pressure away from the cycle and follow the chains to their dead ends (leaf nodes).

Part B: We now show that there is no acceptable labeling for S if the transformation graph G contains a connected component with more than a single cycle. Figure 2.6 shows a set of points with two cycles and no acceptable labeling.

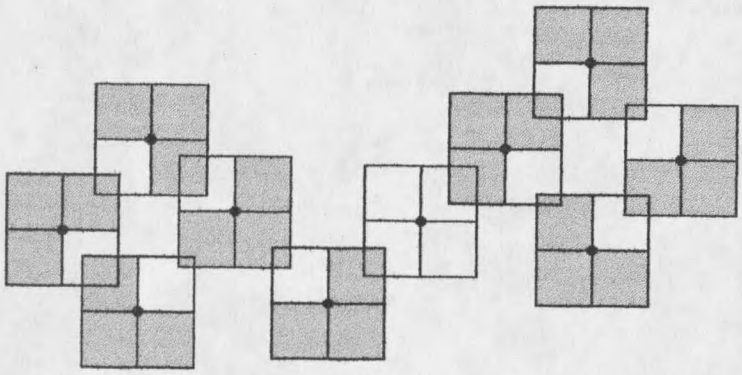


Figure 2.6 A component with multiple cycles has no valid labeling.

It is obvious that due to the nature of a cycle, it must be labeled as described in Part A. Each point in the cycle must accept and exert force on another point in the cycle to obtain a loop. If there are multiple cycles they will each be labeled in this manner. If the two cycles meet at a single point, there will be a violation at this point, as it will be accepting force from two separate points (one from each cycle). If the two cycles are connected with a chain, this will also create a violation. Since every point in the cycle has already received force, we must start at one cycle and exert force away from the cycle. As we continue exerting force down the chain, we will eventually reach the last point in the chain before the second cycle. The point we reach will be receiving force from both the chain and the second cycle and thus not have an acceptable labeling.

Time Complexity

Determining if a given set and edge length has an acceptable labeling appears to take $O(n^2)$ time. This is due to the fact that to create the transformation graph we must check every pair of points to see if there is a sub-square intersection. Remember, however, that we only need to check the at most $24n$ pairs that we found to fall within $2 * D_\infty(S)$ of each other. Therefore, the transformation graph can easily be created in $O(n)$ time. Checking for multiple cycles in $O(n)$ time is trivial using standard graph algorithms such as Breadth-First-Search or Depth-First-Search [BP97].

Candidates for the Optimal Edge Length

We have shown that a given edge length can be verified in linear time. At this point one would be tempted to create an approximation algorithm. We could perform a binary search on the real values between our upper and lower bounds for the optimal edge length to determine an approximate answer within any ϵ value. We can, however, improve this. If we can show that there are only a finite number of possibilities for the optimal edge length then we can run a binary search on the sorted list of optimal edge length candidates. The following lemma assures us that there are a linear number of candidates for the optimal edge length.

Lemma 2.4 *The size of the optimal solution for the MLUST problem under the discrete model must be* $D_\infty(S)$, $d_{min}(p_i, p_j)$, *or* $d_\infty(p_i, p_j)/2$.

The optimal solution, l_{OPT}, implies that if the length were any larger, two sub-squares would intersect at an interior point and cause a valid labeling to no longer exist. If any l

$> l_{OPT}$ will cause an intersection of sub-squares at an interior point, then l_{OPT} must currently be causing two sub-squares to intersect at their boundaries. There are only three possible edge length values for a given pair of points that can cause sub-squares to intersect at their boundaries. The three possibilities are $l = d_\infty(p_i, p_j)$, $l = d_{min}(p_i, p_j)$, and $l = d_\infty(p_i, p_j)/2$. Figure 2.7 illustrates each of these three possible values. It is also clear that since $l_{OPT} \leq D_\infty(S)$ that if $l_{OPT} = d_\infty(p_i, p_j)$, then by the definition of $D_\infty(S)$, it must be true that $d_\infty(p_i, p_j) = D_\infty(S)$. Without taking advantage of earlier properties, this seems to have left us with $2n^2 + 1$ possible values for the optimal edge length. Remember, that there are only at most $24n$ pairs of concern. These are the point pairs within $2 * D_\infty(S)$ of each other. This leaves us with at most $48n + 1$ possible values for the optimal edge length. All candidates for l_{OPT} can be trivially found in linear time.

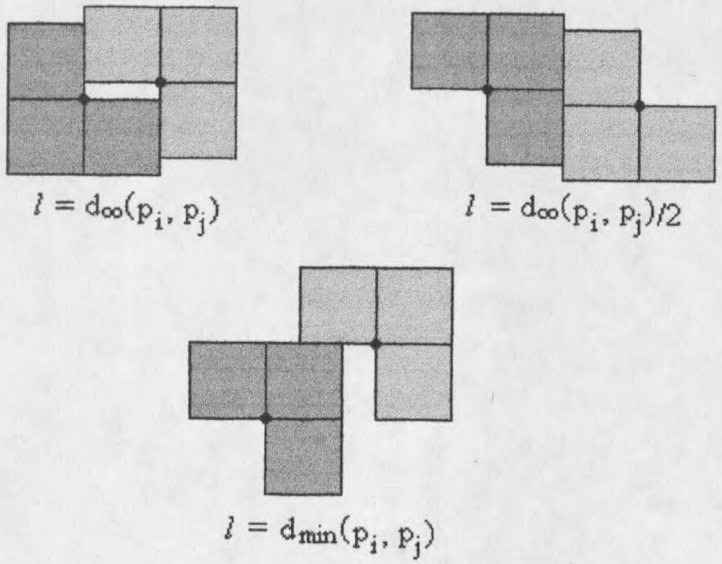


Figure 2.7 Each pair of points contributes at most three candidates for l_{OPT}.

Theorem 2.1 *For any set of points S, finding the largest possible edge length for the MLUST problem under the discrete model can be done in O(n log n) time.*

Proof. Our proof has now been reduced to a series of steps. First, we find $D_\infty(S)$. This, along with our next step, can be completed in $O(n \log n)$ time using a modified version of the divide-and-conquer algorithm for the CLOSEST PAIR problem [PS85]. We detail the algorithm in Chapter 4. Next, find all point pairs (at most $24n$) where $d_\infty(p_i, p_j) \leq 2 * D_\infty(S)$. Once we have obtained a list of points of concern we use the list to determine all candidates for l_{OPT}. We sort the l_{OPT} candidates. And finally, we run a binary search on the l_{OPT} candidates (checking each candidate to determine if the given length is acceptable) to find the largest acceptable value. Our binary search, $O(\log n)$ in conjunction with checking if the edge length is valid takes $O(n \log n)$ time. This and the sorting step are our slowest step which presents us with an optimal $O(n \log n)$ time algorithm for solving MLUST under the discrete model.

CHAPTER 3

MLUST UNDER THE SLIDING MODEL

In this section we present an $O(n^3 \log n)$ time algorithm for the MLUST problem under the sliding model. For the sliding model each square label is placed so that the point feature it labels is located on its boundary. This is in contrast to the discrete model where each square label must be placed so that its corresponding point lies at one of its four corners. The nature of the sliding model, however, does require that at least two of the square labels be placed with their corresponding point at their corners. We call these two square-labels base labels, and the third square-label will be the sliding label. It should be noted that although the sliding label need only be placed with the point on its boundary, it could be placed with the point at a corner.

We follow the same general algorithm as for the discrete model. We begin by showing that for any set and a fixed edge length, we can decide in polynomial time if an acceptable labeling exists. Next, we show that for any set there are a polynomial number of possible candidates for the maximum edge length. And finally, running a binary search on the sorted list of candidate yields the optimal solution in polynomial time.

Determining if an Acceptable Labeling Exists

Unfortunately, for the sliding model we cannot simply have a lemma similar to Lemma 2.1. Unlike the discrete model, there may be a labeling for a series of consecutive points which will break a cycle or chain. We must derive a procedure to

handle this more complicated situation. In the following section we give an example of a series of points that break a cycle.

Cycle Breaker

Assume we have three continuous points in a cycle. In the discrete model force would perpetuate through the cycle from p_{i-1} to p_i and then from p_i to p_{i+1}. In the sliding model, however, it is possible for p_i to be labeled in such a way so as to accept force from p_{i-1} without exerting any force to p_{i+1} (Figure 3.1). This implies that p_i could accept force from both p_{i-1} and p_{i+1} and still have a valid labeling. If such a series of points exist, we call it a *pressure-releasing series*. It should be noted that a pressure-releasing series could be any number of consecutive points. It is clear that this is a different situation from the discrete model where we showed that a point could accept force from no more than one other point.

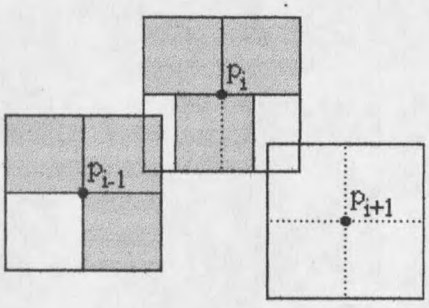


Figure 3.1 A pressure-releasing series

Minimum Labeling

Before we proceed with our labeling procedure, we pause for a definition. A *minimum labeling* is a labeling for a point that minimizes the amount of force it exerts to one of its neighbors, that is, the labeling for a point so that the area of intersection

between its labels and the sub-squares for one of its neighbors is minimized. Assume we have three consecutive points of intersection, p_i, p_j, and p_k. If p_j accepts force from p_i there is a single labeling for p_j that minimizes the amount of force it exerts on p_k. We refer to this as the minimum labeling of p_j with respect to p_k. To create the minimum labeling we first choose the base labels to be the two sub-squares of p_j that are not intersected by either p_i or p_k. Next we choose the sliding label of p_j so that it intersects the force-generating label of p_i at its boundary. Figure 3.2(a) demonstrates a labeling of p_j that is accepting force from p_i and exerting minimum force to p_k. It should be noted that if p_i and p_k are intersecting diagonal sub-squares of p_j (Figure 3.2(b)) then there is no real minimum labeling for p_j with respect to p_k. This is because for p_j to accept the force from p_i then p_j must label the diagonal sub-square from p_i as one of its base labels. Thus any labeling that accepts force from p_i will exert the same amount of force to p_k.

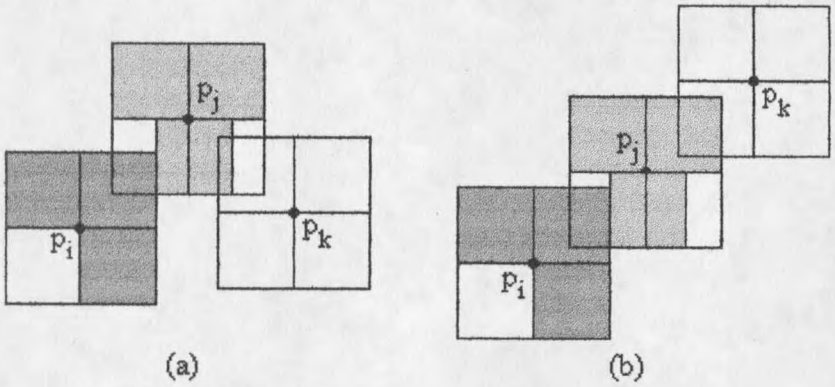


Figure 3.2 Minimum pressure

Starting Position

A *starting node* is a node that has a pre-determined list of labeling configurations, such that if the set has a valid labeling then the starting node can be labeled in at least one

of these pre-determined labeling configurations while still maintaining a valid labeling for the set.

Lemma 3.1 *If a set S has a valid labeling, then there must exist a starting node, p, such that there is a valid labeling for S with p labeled in a manner that minimizes the pressure it has with respect to one of its unlabeled neighbors.*

The minimum force explained earlier shows that for a point receiving force we only need to be concern with O(1) possible labeling configurations. With our initial point, however, this is not true. Our initial point is not receiving force from any other point and does not have prior knowledge about the force that will be generated to it. It is clear from Figure 3.3 that the point p_i has only one possible labeling that will allow the set to have a valid labeling. If we were to start at point p_i we would not be able to determine this labeling. It is not exerting minimum force to any of its unlabeled neighbors. How then can we start at p_i, since it has an infinite number of possible labeling configurations? The answer is we cannot. Note, however that at each end of this pressure-releasing series there are points that are labeled in minimum force manner. That is they are labeled in a manner as to generate minimum force to one of their unlabeled neighbors. If we could find one such point then we could choose that point as our starting position.

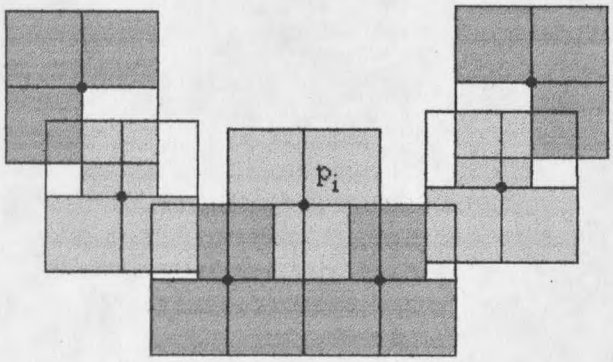


Figure 3.3 A point, p_i, that cannot be a starting point.

In searching for a starting position we cannot simply choose the end point of a pressure-releasing series, as it might be a member of another series of pressure-releasing points. We note, however, that a pressure releasing series will always lie in a vertical or horizontal manner. At each end of a horizontal series of points we may run into a vertical series, and at each end of the vertical series we may likewise run into a horizontal series. It is obvious that with a finite set of points we must have either an end point or a corner. At either an end point or a corner will be an acceptable starting node. Thus, although we cannot find a starting node by testing a constant number of nodes, we know that if a valid labeling exists then the set must contain at least one acceptable starting node. We therefore rely on this fact and try each point as a starting node. We end when we have found a valid labeling for the set or have exhausted all points as starting nodes.

Labeling Procedure

We now discuss the actual procedure used to determine if an acceptable labeling exists for a given set and edge length. The limited branching procedure follows the same general format that was used by A. Shamir et al. in [EIS76] to solve problems resembling the 2-SAT problem.

We begin by selecting a point, p_i, as our initial position and marking the node as the *decision* node. Next, we choose a minimum labeling for p_i with respect to one of its neighbors. We then traverse the graph following the minimum force that must be generated from our current labeling. At each point with a single minimum labeling, we label the point, mark the point as *temporary*, and continue to follow any force. If we reach a point p_j that has multiple minimum-labeling configurations then we exert

minimum force to each of its neighbors. That is we choose each neighbor, p_k, and we determine the minimum labeling of p_j with respect to p_k while still withstanding the incoming force. We then exert this determined force to p_k and continue. If we reach a point that has no acceptable labeling then our last decision (the labeling configuration made at p_i) is unacceptable. We backtrack to p_i and remove all temporary nodes. We then choose a different minimum labeling for p_i and repeat the process. If, after choosing some minimum labeling for p_i and following all force, we reach a state such that no point has been labeled in an unacceptable manner, then we proceed to the next decision node. That is we mark p_i and all temporary nodes as permanent, move forward to p_j, and mark it as our new decision node. If after continuing with the procedure we find ourselves at a state in which all nodes have been labeled in an acceptable manner, then stop, an acceptable labeling exists. If after trying all minimum-labeling configurations for a decision node without success, then no acceptable labeling exists for the chosen starting point. We then choose a different starting node and repeat the entire process. If after trying each node as our starting node, no acceptable labeling is found, then no acceptable labeling exists for the given set and edge length.

Correctness of Our Labeling Procedure

It is not directly apparent that our algorithm always finds a valid labeling configuration, if one exists. We now attempt to explain the correctness of the previous procedure. To be confident in our algorithm we must show that if a set has a valid labeling then at any decision node a valid choice exists. It is obvious that if at each

decision node we have a valid choice, we will eventually end when every decision node has been decided and the entire set has been labeled correctly.

At each decision node (excluding the starting point) we have a binary decision, that is, there are only two possible minimum-labeling configurations. In Figure 3.4, point p_j is our decision node. It is accepting force from the previously labeled p_i and will either be labeled with a minimum labeling with respect to p_k (Figure 3.4(a)) or with respect to p_l (Figure 3.4(b)).

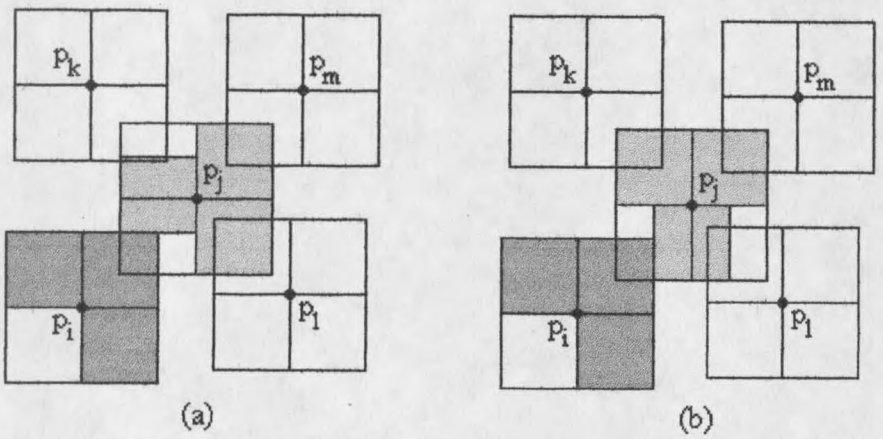


Figure 3.4 Each decision node is binary.

Whenever minimum force is exerted in one direction, maximum force is generated in the opposing direction. We can therefore just as easily view the choice as in which direction we exert maximum force.

It is also important to understand that each decision must be made regardless of the incoming pressure, that is, in Figure 3.4 maximum force will be exerted to p_k or p_l independent of the incoming force from p_i. Although the incoming force does not affect the maximum force that will be generated, it is this incoming force from p_i that will determine the extent of the minimum force to be exerted in the alternate direction.

Hence, if there was no incoming force from p_i due to earlier decisions then we would still need to send maximum force in one direction, but we would exert no force in the other direction. Because of the incoming force from p_i we must exert some minimum force in the alternate direction that we send maximum force.

Assume set S has a valid labeling and we are at decision node p_i. Assume also that we choose to exert maximum force in direction A and, after following the subsequent force, we arrive at a point, p_j, that has already been added to our permanent list due to an earlier decision. It is impossible at this point for force to be exerted to p_j that will cause an invalid labeling. This is because if a path from p_i to p_j exists such that force generated from p_i will cause an invalid labeling of p_j, then when p_j was originally labeled it would have generate force to p_i and p_i could not be our current decision node. This implies that if a choice is invalid it can only be invalid due to points that have not already been labeled. Therefore previous decisions have no effect on whether our current decision node has a valid choice. If the set S has an acceptable labeling then our current decision node must have a valid choice.

Time Complexity

The procedure has a time complexity of $O(n^3)$. The important step in creating a polynomial time algorithm is that after each decision we enter it into our permanent list where it will never be changed. It is obvious that for a given starting point each node may only be a decision node once. It is also true that with each subsequent decision the minimum force to check for validity will take at most $O(n)$ time. Each node has O(1)

decisions (actually only two) and thus the algorithm will run in $O(n^2)$ time for a given starting position and $O(n^3)$ time to try each node as the starting position.

Lemma 3.2 *Given a set S and edge length l, it can be determined in $O(n^3 \log n)$ time if an acceptable labeling exists.*

Candidates for the Optimal Edge Length

We now show that the number of possible candidates is $O(n^3)$. As with the discrete model one possibility for l_{OPT} is $D_\infty(S)$. It also remains true that for each pair of points $d_{min}(p_i, p_j)$ is a candidate for l_{OPT} (For an explanation of why these distances are possibilities for l_{OPT}, refer to Chapter 2). Unlike the discrete model, however, we have several additional possibilities. Due to the ability of the sliding label, there can be any number of consecutive points whose sliding labels intersect at their boundaries. These consecutive points will have two end points, p_i and p_j, which are $d_\infty(p_i, p_j)$ apart (Figure 3.5). We can no longer be satisfied to check only the points within $2 * D_\infty(S)$ of each other. We must now check every pair of points, which is $O(n^2)$. And for each pair of points we must check $d_\infty(p_i, p_j)/k$, where $l \leq k \leq n$. The result is $O(n^3)$ possible candidates for l_{OPT} and a time complexity of $O(n^3)$.

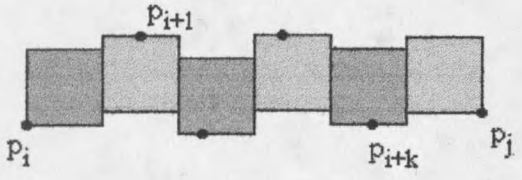


Figure 3.5 A series of consecutive labels intersecting at their boundaries.

Lemma 3.3 *The size of the optimal edge length for the MLUST problem under the sliding model must be $D_\infty(S)$, $d_{min}(p_i, p_j)$, or $d_\infty(p_i, p_j)/k$, where $1 \leq k \leq n$.*

We now have all the pieces necessary to present our theorem for the MLUST problem under the sliding model.

Theorem 3.1 *For any set of points S, finding the largest possible edge length for the MLUST problem under the sliding model can be done in $O(n^3 \log n)$ time.*

Once again our proof has been reduced to several simple steps. First, we find $D_\infty(S)$, which can easily be done in $O(n \log n)$ time as described in Chapter 3. Next, we find the distances $d_{min}(p_i, p_j)$ and $d_\infty(p_i, p_j)/k$, for $l \leq k \leq n$. These are the candidates for the optimal edge length and can be found in $O(n^3)$ time. We then sort the candidates for l_{OPT}, which will be our slowest step and take $O(n^3 \log n)$ time. And finally, we run a binary search on the candidates for l_{OPT}, finding the true value of l_{OPT}.

CHAPTER 4

IMPLEMENTATION

A practical solution in terms of implementation is not always easy to obtain directly from a theoretical solution. For the MLUST problem we are fortunate to find our solution can be implemented with minimal changes.

Upper Bound

In an implementation of the MLUST problem it is quite natural that we will be concerned with borders. Certainly in labeling a map we do not want any labels to be placed in such a manner that a portion of the label is outside of the viewing range. To avoid a label being placed in this manner we must calculate the shortest distance between a point and an edge. We will denote this distance as NE (nearest edge). It is obvious that for an acceptable labeling to exist, the edge length of our square labels cannot exceed NE. Therefore, our upper bound for l_{OPT} will be min(NE, $D_\infty(S)$). The following example shows an implementation of the MLUST problem calculating the range for l_{OPT}.

Figure 4.1 shows the result of calculating $D_\infty(S)$. The two points that are closest to each other have each been given a small halo. Their sub-squares have also been display so that one can visually see why this is an upper bound for l_{OPT}. This distance of 43 units is an upper bound for the set of points.

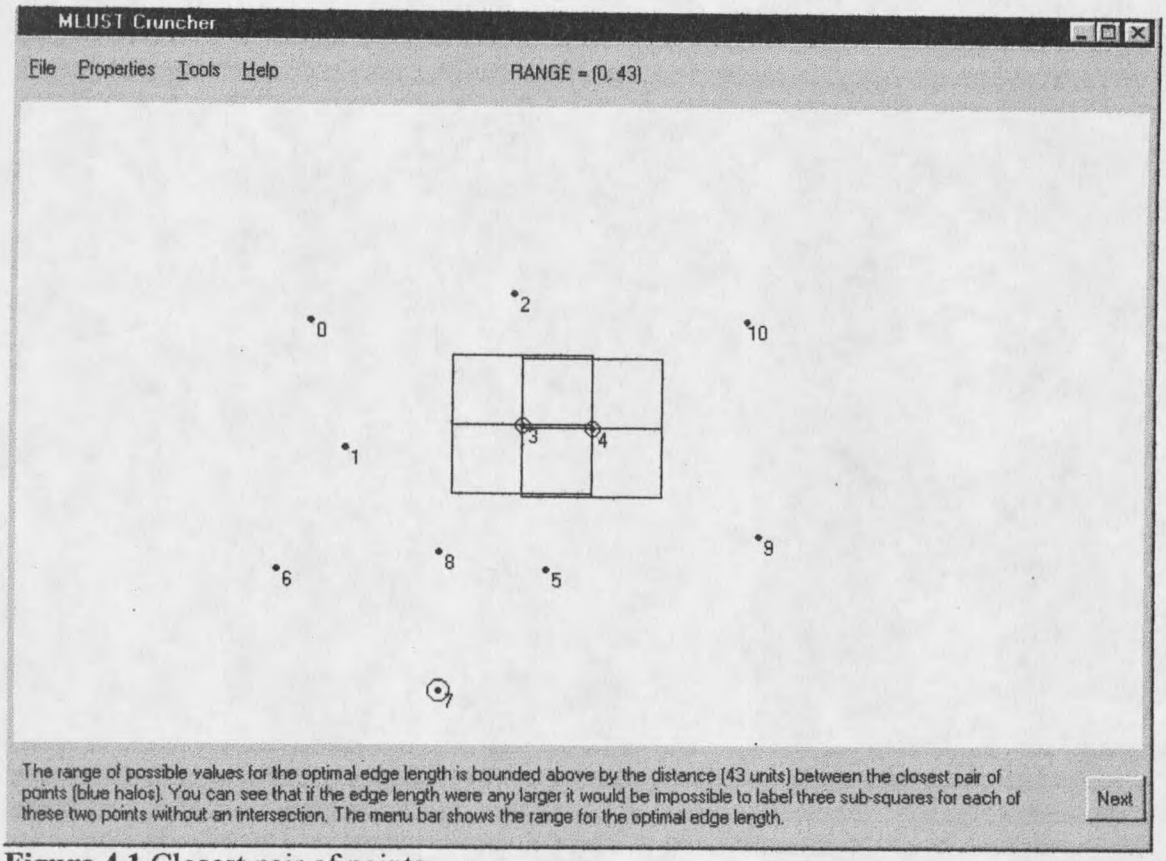


Figure 4.1 Closest pair of points.

Figure 4.2 shows the result of calculating NE. The point that is closest to an edge has been given a large halo and is displaying its sub-squares for the edge length NE. It is obvious that if the sub-squares were any larger, it would be impossible to label three of its sub-squares without having at least one label outside of the viewing area. Since this distance of 33 units is smaller than $D_\infty(S)$ (43 units) the new upper bound for this set of points is updated to 33 units.

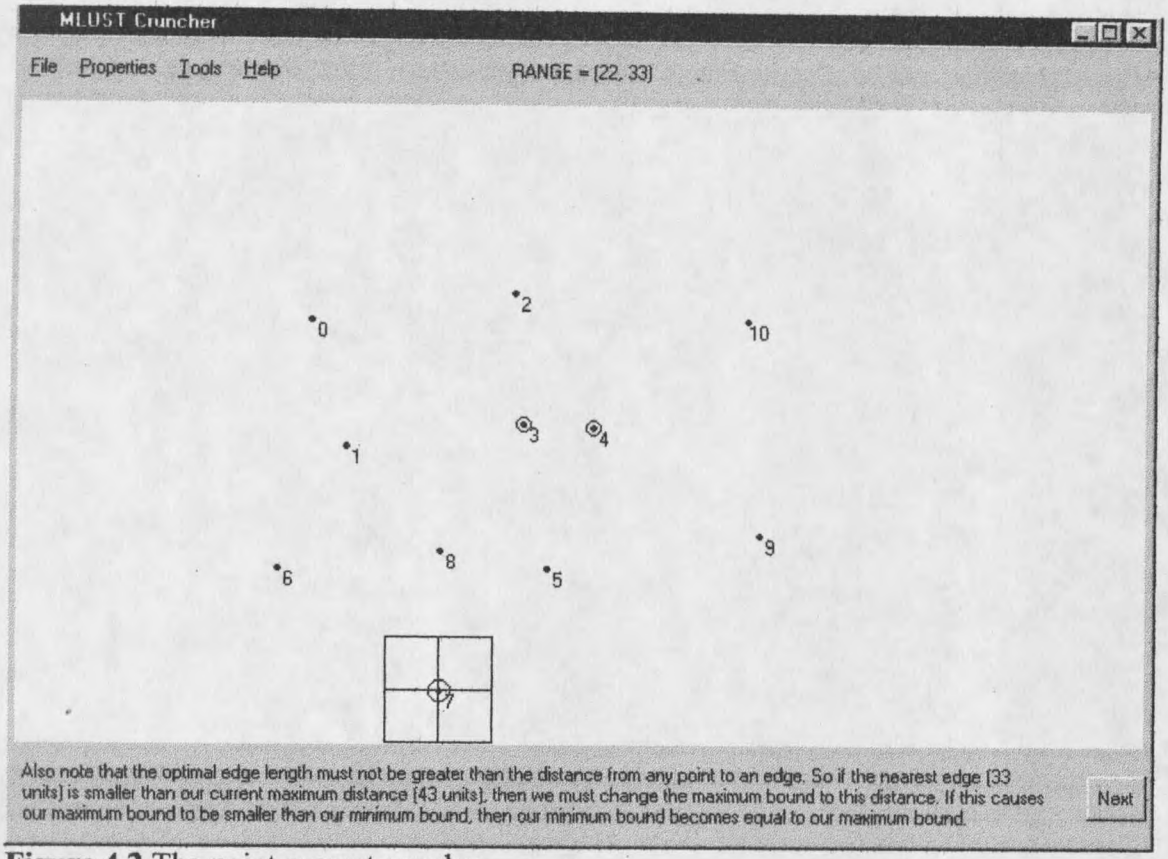


Figure 4.2 The point nearest an edge.

It should also be noted that if NE is smaller than $D_\infty(S)/2$ (lower bound) then we must update the lower bound to be NE. This would imply NE = l_{OPT}.

Finding $D_\infty(S)$ and Pairs of Concern

Remember that from our earlier discussion of the MLUST problem, under the discrete model, we are only concerned with point pairs within the upper bound for l_{OPT} of each other. The algorithm described in this section is used to find all values that are within a certain range, R, of each other under the L_∞-metric. Whenever we talk about distances in this section we are referring to the distance under the L_∞-metric. The

algorithm has a time complexity of optimal O(n log n). It is a modified version of the divide-and-conquer algorithm explained in [PS85] for finding the CLOSEST PAIR in a 2D set. It assumes pre-processing includes generating a list of the points sorted according to their x-coordinate. The algorithm is as follows:

1. Divide the set S into two equal size sets, S_1 and S_2, according to the vertical median line, M.
2. Find all point pairs within R of each other in S_1 and S_2 recursively.
3. Let P_1 be the set of points in S_1 that are within R of the dividing line M. Let P_2 be the corresponding subset of S_2 (Figure 4.3). Project P_1 and P_2 onto M and sort them according to their y-coordinates to obtain the sorted lists P_1* and P_2* respectively.
4. Search for any pair with one of the points in P_1 and the other point in P_2 that are within R of each other. We find these pairs with the following technique. Scan each point in P_1*. At each point, p_i, in P_1* calculate the distance from p_i to any point, p_j, in P_2* that is within R of p_i in the y-direction. If $d_\infty(p_i, p_j) \leq R$ the pair is included.

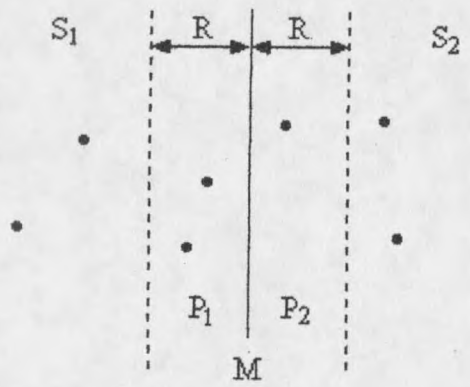


Figure 4.3 Points within R of the dividing line.

Note: As we scan each point in P_1*, the pointer for P_2* may move up and down, but will never need to go beyond R in either direction. There can be at most 15 points in P_2* within R of any point in P_1* (Figure 4.4). The algorithm is easily accomplished in O(n log n) time.

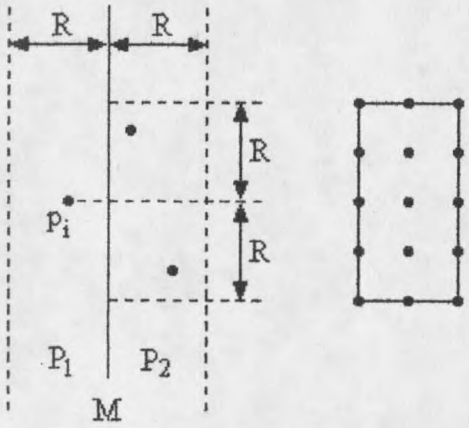


Figure 4.4 The maximum number of points within R of p_i.

The algorithm described is also used for computing $D_\infty(S)$. The only modification is that R becomes a variable that holds the distance between the current closest pair.

Finding the Optimal or Approximate Solution

In our algorithm we find all possible values for l_{OPT} and store them. For the discrete model this is easily done in $O(n)$ time with $O(n)$ storage. Under the sliding model, however, it takes $O(n^3)$ time and $O(n^3)$ storage. For both models the slowest step for our algorithm is sorting this list of candidates. In many cases it may be to our advantage to approximate the solution and avoid creating and sorting this list. We can easily approximate the answer with a binary search using the upper and lower bounds for the optimal edge length as our initial upper and lower values. Looping through our binary search k times yields us a solution that is $k/(k+1)$ of the optimal solution. In other words, it will guarantee a solution within ε of the optimal solution where $k = \log_2 (D_\infty(S)/\varepsilon) - 1$.

It is also important to note that in most graphical visualizations, we will be dealing with pixels. In this case, there is no advantage in finding the candidates for the optimal edge length. In a graphical display that uses pixels, all distances must be an integer value. Therefore, there are a finite number of possibilities for the optimal edge length, and thus the optimal edge length must be an integer value between the upper and lower bounds. In Figure 4.5 we have successfully labeled the set of points with a distance of 36 units. The upper bound for the set is 48 units and the lower bound for the set is 24 units. Our graphical display limits us to using integer distances and we can simply run a binary search on all integers between the upper and lower bounds to obtain the optimal edge length.

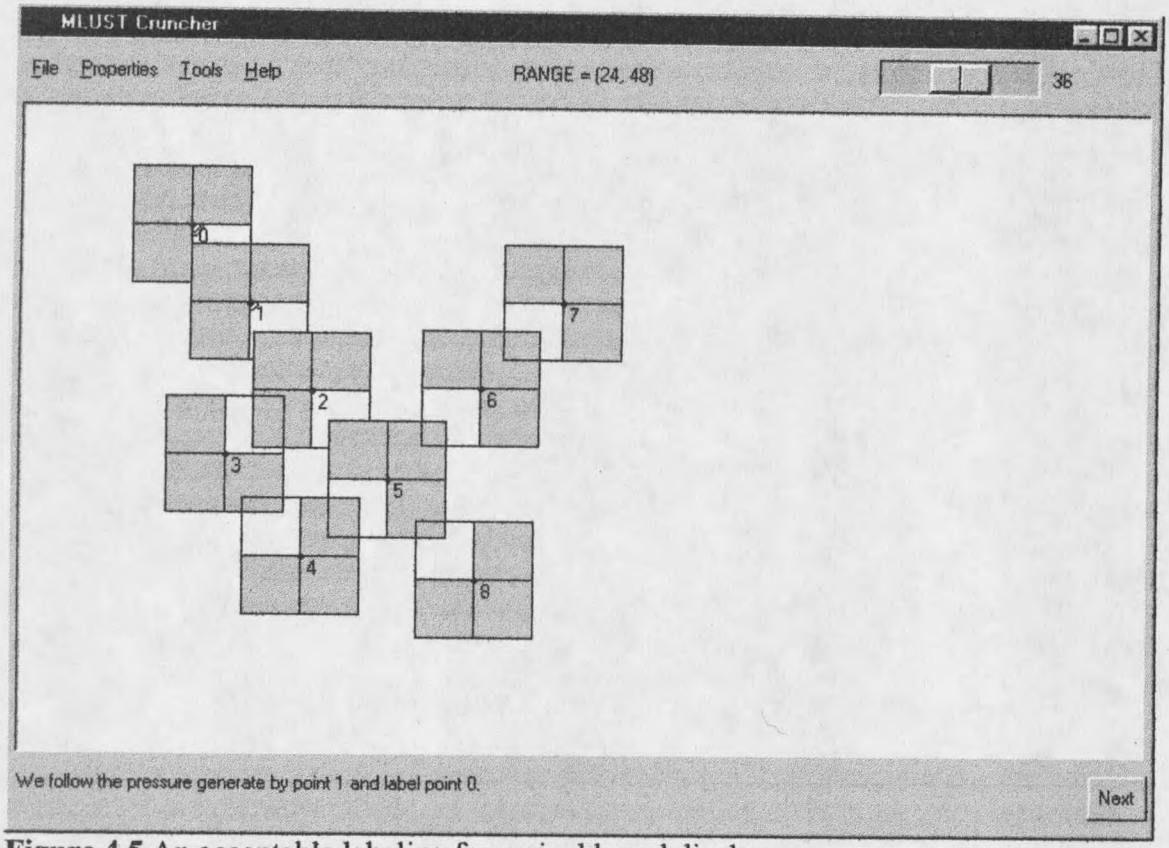


Figure 4.5 An acceptable labeling for a pixel based display.

CHAPTER 5

CONCLUDING REMARKS

There is still much to be done in the area of point feature labeling and more specifically in the area of multi-label point feature labeling. Can the gap between the $O(n^3 \log n)$ upper bound and the $\Omega(n \log n)$ lower bound for the sliding model be reduced? What improvements could be made to lower the time complexity for the algorithm presented for the sliding model in this thesis?

Acknowledgments

I thank my advisor, Dr. Binhai Zhu, for his guidance in my research and patience with my numerous questions.

REFERENCES CITED

[B96] B. Chazelle et al. "Application Challenges to Computational Geometry," *Computational Geometry Impact Task Force*. Technical Report TR-521-96, 1996

[BP97] K. Berman and J. Paul, "Fundamentals of Sequential and Parallel Algorithms," *PWS Publishing Company*, pages 292-303, 1997.

[CMS95] J. Christensen, J. Marks, and S. Shieber, "An Empirical Study of Algorithms for Point-Feature Label Placement," *ACM Transactions on Graphics*, 1995.

[DMM00] S. Doddi, M. Marathe, and B. Moret, "Point Set Labeling with Specified Posistions," *Proc. 16th Annual ACM Symposium on Computational Geometry*, pages 182-190, 2000.

[DMMMZ97] S. Doddi, M. Marathe, A. Mirzaian, B. Moret, and B. Zhu, "Map Labeling and its Generalization," *Proc. 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 148-157, 1997.

[EIS76] S. Even, A. Itai, and A. Shamir, "On The Complexity of Timetable and Multicommodity Flow Problems," *Siam Journal of Computing*, Vol. 5, No. 4, December, 1976.

[FW91] M. Formann and F. Wagner, "A Packing Problem With Applications To Lettering Maps," *Proc. 7th Annual ACM Symposium on Computational Geometry*, 1991

[GJ79] M. Garey and D. Johnson, "A Guide to the Theory of NP-Completeness," *W.H. Freeman and Company*, 1979.

[I75] E. Imhof, "Positioning names on Maps," *The American Cartographer*, pages 128-144, 1975.

[KSWW01] V. Kapoor, T. Strijk, F. Wagner, and A. Wolff, "Three Rules Suffice for Good Label Placement," *Algorithmica Special Issue on GIS*, 2001.

[KSW99] M. van Kreveld, T. Strijk, and A. Wolff, "Point Set Labeling With Sliding Labels," *Computational Geometry Theory and Applications*, 13, pages 21-47, 1999.

[KT98a] K. Kakoulis and I. Tollis, "A Unified Approach to Labeling Graphical Features," *In Proc. 14th Annual ACM Symposium Computational Geometry* (SoCG'98), pages 347-356, 1998.

[KT98b] K. Kakoulis and I. Tollis, "On the Multiple Label Placement Problem," *In Proc. 10th Canadian Conf. Computational Geometry* (CCCG'98), pages 66-67, 1998.

[MS91] J. Marks and S. Shieber, "The Computational Complexity of Cartographic Label Placement," *Technical Report TR-05-91*, Harvard CS, 1991.

[PS85] F. Preparata and M. Shamos, "Computational Geometry: An Introduction," *Springer-Verlag*, 1985.

[QWXZ00] Z. P. Qin, A. Wolff, Y. Xu, and B. Zhu, "New Algorithms for Two-label Point Labeling," *Proc. 8th European Symposium on Algorithms* (ESA'00), pages 368-379, Springer-Verlag, LNCS series 1879, 2000.

[SY82] J. M. Steele and A. C. Yao, "Lower Bounds for Algebraic Decision Trees," *Journal of Algorithms*, 1982.

[W94] F. Wagner, "Approximate Map Labeling Is In Omega (n log n)," *Information Processing Letters*, 52(3): pages 161-165, 1994.

[WW95] F. Wagner and A. Wolff, "Map Labeling Heuristics: Provably Good and Practically Useful," *In Proc. 11th Annual ACM Symposium on Computational Geometry* (SoCG'95), pages 109-118, 1995.

[Y72] P. Yoeli, "The Logic of Automated Map Lettering," *The Cartographic Journal*, 9(2), pages 99-108, 1972.

[ZP99] B. Zhu and C. K. Poon, "Efficient Approximation Algorithms for Multi-label Map Labeling," *Proc. 10th International Symposium on Algorithms and Computation* (ISAAC'99), 1999.

[ZQ00] B. Zhu and Z. P. Qin, "New Approximation Algorithms for Map Labeling with Sliding Labels," *Journal of Combinatorial Optimization*, 2000.