A Note on Three-Label Point Labeling

Antoine Vigneron*

August 30, 2001

Abstract

Let P be a set of n points in the plane. We want to find a set $S(P,l^*)$ of axis–parallel squares with edge length l^* such that each point of P is associated with three squares, each point of P lies on the boundary of its three associated squares, no two squares of $S(P,l^*)$ intersect, and l^* is maximized. We show how to solve this problem in $O(n^2 \log n)$ time.

1 Introduction

Recently Duncan et al. [3] gave an algorithm for labeling a set of n points with triples of sliding squares. We modify their approach and bring down the running time bound from $O(n^3 \log n)$ to $O(n^2 \log n)$, obtaining the fastest algorithm known so far. We note that the whole algorithm is simple and implementable, and we expect it to perform well in practice. Duncan et al. [3] also give applications of map labeling and summarize the state of the art.

Several variants of three-label point labeling have been considered. Given a set P of n points in the plane and a length l, the *decision* problem is to find a set $S(P,l)$ of squares of edge length l that are non intersecting, and such that each point of P is associated with three squares adjacent to it. l is called the *size* of the solution. In the *discrete* model, a point is a vertex of each associated square, whereas in the *sliding* model it may lie on the side of one of these

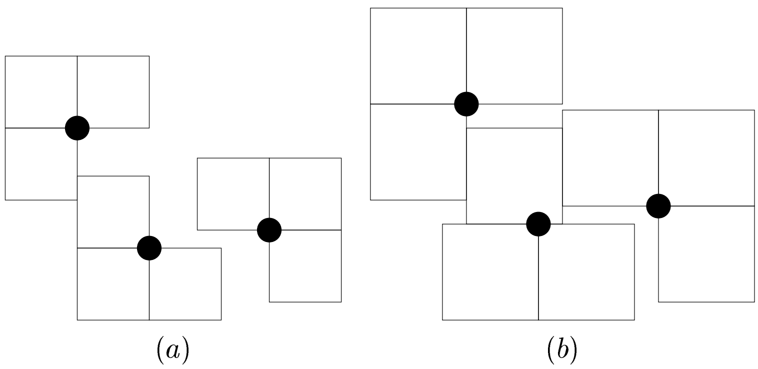

Figure 1: Optimal solution under the discrete model (a) and the sliding model (b) for the same set of three points.

squares (see Fig. 1). The *optimization* problem is to find $S(P,l^*)$ where l^* is maximum. In this note we are interested in the optimization problem under the sliding model.

*The Hong Kong University of Science and Technology.

2 Preliminaries

Duncan et al. [3] give an optimal $O(n \log n)$ time algorithm for the discrete optimization problem. Here we cite two other results they prove under the sliding model.

Lemma 1 *The decision problem with parameters (P, l) can be solved in $O(n^2)$ time.*

For two points $p = (x, y)$ and $p' = (x', y')$ of P, we denote by $d_\infty(p, p')$ the maximum of $|x - x'|$ and $|y - y'|$.

Lemma 2 *When the sizes of the solutions under the discrete and the sliding model differ, then the size l^* of the solution under the sliding model is of the form $l^* = d_\infty(p, p')/k$ where $(p, p') \in P^2$ and $k \in \{1, 2, \ldots n - 1\}$.* [1]

Duncan et al. [3] construct a sorted list of all the possible values of l^* given by Lemma 2, it takes $O(n^3 \log n)$ time which dominates the overall running time. Then they perform a binary search on this list which, using Lemma 1, only takes $O(n^2 \log n)$ time. In this note we show that the search can be performed efficiently, without enumerating all the possible values of l^*, by a decimation argument. The main idea is reminiscent of Blum et al. [1] algorithm for finding a median in linear time.

Let E be a finite set of pairs of real numbers. We denote $W = \sum_{(x,w) \in E} w$ and $W_{<y} = \sum_{x<y,(x,w) \in E} w$. Similarly $W_{>y} = \sum_{x>y,(x,w) \in E} w$. A weighted median m_E of E is such that $(m_E, w_E) \in E$, $W_{<m_E} \leq W/2$ and $W_{>m_E} \leq W/2$. A weighted median can be computed in $O(|E|)$ time (see the book by Cormen et al. [2] page 193).

3 Algorithm

The set of the distances $d_\infty(p, p')$ for all $(p, p') \in P^2$ is denoted by D. Initially, the search interval (l_1, l_2) is $(0, \infty)$. For all $d \in D$, we denote by L_d the set of the lengths d/k that belong to (l_1, l_2) and such that $k \in \{1, 2, \ldots n - 1\}$. We denote by x_d the median of L_d and w_d is the cardinality of L_d. We can find both these values in $O(1)$ time without computing L_d explicitly, for instance in case L_d is non–empty the index k associated with its smallest element is given by $\min(n - 1, \lceil \frac{d}{l_1} - 1 \rceil)$. The recursive function described below reduces the search interval (l_1, l_2) and returns the optimal size when it differs from the solution to the discrete problem.

Algorithm *decimate*(l_1, l_2)
compute $E = \{(x_d, w_d) \mid d \in D\}$
if $W = 0$
  then return l_1
  else compute the weighted median m_E
    if the sliding decision problem (P, m_E) has a solution
      then return *decimate*(m_E, l_2)
      else return *decimate*(l_1, m_E)

Correctness of this algorithm follows from Lemma 2 and the following invariant: the decision problem with parameters (P, l_1) has a solution and the one with parameters (P, l_2) has no solution. Now we prove that it runs in $O(n^2 \log n)$ time. First note that each call to *decimate*,

[1] In the article by Duncan et al. [3], k cannot take the value $n - 1$, it seems to be a misstatement as can be seen in Fig. 1(b).

ignoring the recursive calls, takes $O(n^2)$ time. Indeed, step 1 can be performed in constant time per $d \in D$ as was explained in the previous paragraph, and step 5 can be performed in $O(n^2)$ time by Lemma 1. So we only need to prove that a constant fraction of $L = \bigcup_{d \in D} L_d$ is discarded at each recursive call.

Lemma 3 *At each recursive call to decimate, the cardinality of L shrinks by a factor at least 4/3.*

Proof: Note that $|L| = W$. We show that $|L \cap (l_1, m_E)| \leq 3W/4$, the proof that $|L \cap (m_E, l_2)| \leq 3W/4$ is similar. For all $d \in D$ such that $x_d \geq m_E$, the at least $w_d/2$ lengths in L_d that are greater or equal to x_d are also greater or equal to m_e, therefore they do not appear in $L \cap (l_1, m_E)$. So $|L \cap (l_1, m_E)| \leq W - \frac{1}{2}\sum_{x_d \geq m_E} w_d$. So by definition of the weighted median $|L \cap (l_1, m_E)| \leq W - \frac{1}{4}W$. $\square$

Putting everything together, we reach the conclusion:

Theorem 1 *The optimization problem under the sliding model can be solved in $O(n^2 \log n)$ time.*

Proof: First we solve the problem under the discrete model in $O(n \log n)$ time [3] and obtain an optimal size l_0^*. Then we compute $l_1^* = \mathit{decimate}(0, \infty)$, it runs in $O(n^2)$ time per level of recursion, and it recurses $O(\log n)$ times by Lemma 3, so the whole process takes $O(n^2 \log n)$ time. By lemma 2, the maximum of l_0^* and l_1^* is l^*. We obtain the set $S(P, l^*)$ by Lemma 1 in $O(n^2)$ time. $\square$

4 Conclusion

We note that a simpler randomized algorithm achieves the same time bound. The idea is to pick an element l_m of L uniformly at random. This can be done in $O(n^2)$ time using the fact that for all $d \in D$, the cardinality of L_d can be found in constant time. The only difference with our deterministic algorithm is that l_m plays the role of m_E.

The main open question now is whether a near–linear time algorithm can be designed. In order to obtain a sub–quadratic time bound, a very different method would have to be discovered that does not make use of the current $O(n^2)$ time decision algorithm.

References

[1] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *J. of Computer and System Sciences*, 7(4):448, 1972.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms.* MIT Press and McGraw-Hill Book Company, 6th edition, 1992.

[3] Rob Duncan, Jianbo Qian, and Binhai Zhu. Polynomial time algorithms for three–label point labeling. In *Proc. 7th Ann. Internat. Conf. Computing and Combinatorics*, volume 2108 of *Lecture Notes Comput. Sci.*, pages 191–200. Springer-Verlag, 2001.