A Fast Algorithm for Updating a Labeling to Avoid a Moving Point*

Farshad Rostamabadi† Mohammad Ghodsi‡

Computer Engineering Department, Sharif University of Technology, Tehran, IRAN

IPM School of Computer Science, Tehran, IRAN

Abstract

Given a set of labeled points forming a valid map labeling, we are interested in a fast update of the labels if a point shaped object moves on an unknown path in the map. In this paper, there are n labels that assumed to be axis-parallel, unit-length, and square-shaped, each attached to one point in the middle of one of its edges. We assume that a moving object can freely move on the map and sends notifications about its new positions. An updated labeling should include all labels with no overlaps, avoid the current position of the moving point, and use labels with length close to unit-size as possible. The existing algorithm for this problem runs in $O(n \lg n)$ per each position notification. We present an algorithm that needs a preprocessing of $O(n^2)$ time, but can update the map, for any new position of the moving point, in $O(\lg n + k)$, where $k \leq 2n$ is the minimum number of update operations needed.

1 Introduction

Automated label placement is an important problem in map generation, geographical information systems, and computer graphics. This problem, in its simple form, is to attach a label (regularly a text) to each point, line, curve, or a region in the map. Point-label placement has received good attention. In a valid labeling, labels should be pairwise disjoint, and each label should be attached to its feature point [1]. There are different variations of point-labeling that are discussed in [2, 3, 4, 5, 6].

In this paper, we are interested in a fast update of labels in a point-labeling map. For simplicity, we assume that our map is composed of a number of points (n) each labeled by a unit-length axis-parallel square label. The point of each label appears in the middle of one of its edges. Maps with more general labeling can also be considered in our algorithm.

The labeling should be updated when the point-shaped object moves on an unknown path in the map. We assume that the object can freely move on the map, and we are only notified when its position is changed (like a mouse movements on screen). The new labeling should be valid in a way that all points preserve their labels, but the labels may have to flip or resize to avoid the current position of the moving point. Besides, our goals is to have labels with lengths as close to one (unit length) as possible, and to use the fewest number of label flip operations (according to the previous updated map).

Since each position notification of the moving object implicitly means that the object is removed from its old position, we can see each notification event as follows. First, the object is removed from its old position and the labeling is set back to its initial form. Second, the object is inserted into its new position and the problem is to find a new optimal labeling defined as above. This view of the problem, greatly helps us find the optimal solution in the optimal time.

Applications for this problem can be found in computer graphics, computer games, flight animation, and in other related fields.

We show that given the original map, we can create several data structures in a preprocessing phase that use $O(n)$ space and $O(n^2)$ time, so that for any position of the moving point, the updated labeling with the mentioned optimum property can be found in $O(\lg n + k)$. Here, $O(\lg n)$ is used for point location and $k \leq 2n$ is the smallest number of flip and resize operations needed.

The existing solution for this problem is based on 2-SAT algorithm. This solution is independent of the existing valid labeling, and should be found for each position of the moving object from scratch. Details of this reduction can be found in [7].

2 Definitions and the General Idea

The problem is precisely defined as follows. We are given a valid labeling L composed of points $\mathcal{P} = \{p_1, p_2, \dots, p_n\}$ and unit-length axis-parallel square labels $\mathcal{L} = \{\ell_1, \ell_2, \dots, \ell_n\}$, where ℓ_i is attached to p_i on the midpoint of one of its edges. For each position of the moving point q, the problem is to update L and obtain a new q-*avoiding* labeling L_q for all points in $\mathcal{P}$, such that q does not intersect with any label in L_q, while the new label sizes are as close to one as possible. We also want to create L_q with the minimum number of operations. Such a final re-labeling is denoted by q-*avoiding optimum labeling*.

We define the operations more precisely as follows. A label ℓ_i can be flipped over the edge containing its corresponding point p_i. ℓ_i can also be resized to any length $\gamma \leq 1$ as

*This work has been supported by a grant from IPM School of CS (No. CS-1382-2-02).

†farshad@mehr.sharif.edu

‡ghodsi@sharif.edu

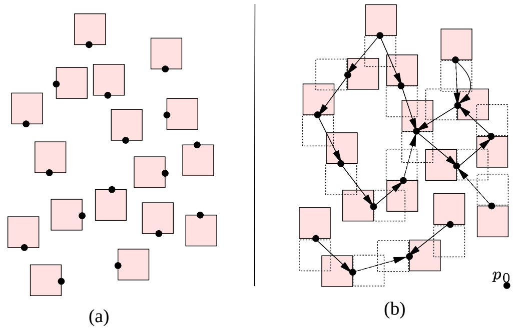

Figure 1: (a) Initial labeled map. (b) The conflict graph: domino edges (solid) and blocking edges (dashed). Edges ending at p_0 are not shown.

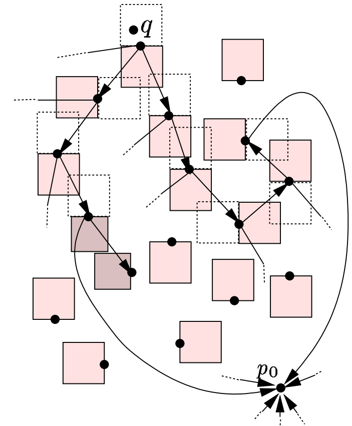

Figure 2: G_q^α and the optimal generated L_q^α.

long as p_i remains on the mid-point of its edge.

Conflict graph $\mathcal{G} = (\mathcal{P} \cup \{p_0\}, \mathcal{E})$ is defined as a weighted directed multi-graph on the set of points $\mathcal{P}$ and a dummy vertex p_0 (this dummy vertex is required to model flip operation for vertices with no outgoing domino edge), to model all possible flip and resize operations on $\mathcal{L}$. There are two different edges in this graph: One representing the normal flip operations, called *DominoEdges* to convey the domino effect that may be caused by a flip. We will show that a flipped label can not flip any further. But, there may be cases where two flipped labels overlap, and we should resize one or both to avoid label overlaps. These operations are modeled by *BlockingEdges*. These edges are precisely defined as,

$$\begin{aligned} \textit{DominoEdges} &= \{(p_i,p_j) | f(\ell_i) \cap \ell_j \neq \emptyset\} \cup \\ &\quad \{(p_i,p_0) |\ \forall p_i \in V - \{p_0\}\}, \text{and} \\ \textit{BlockingEdges} &= \{(p_i,p_j) | f(\ell_i) \cap f(\ell_j) \neq \emptyset\}. \end{aligned}$$

The resize operation is needed when two (original or flipped) labels, say ℓ_a and ℓ_b, overlap. There are many resize values of γ_a and γ_b such that the resized labels $r(\ell_a, \gamma_a)$ and $r(\ell_b, \gamma_b)$ do not overlap. From the problem definition, we are interested in the values where $\min\{\gamma_a, \gamma_b\}$ is maximized. We define this value as $g(\ell_a, \ell_b)$.

Based on the above g function, we define a weight function $w(e)$ for each edge $e = (p_i, p_j) \in \mathcal{E}$ as

$$w(e) = \begin{cases} g(f(\ell_i), \ell_j) & \text{if } (p_i,p_j) \in \textit{DominoEdges}, \\ g(f(\ell_i), f(\ell_j)) & \text{if } (p_i,p_j) \in \textit{BlockingEdges}, \\ 1 & \text{if } p_j = p_0. \end{cases}$$

A sample labeled map is shown in Fig. 1(a). The corresponding conflict graph is shown in Fig. 1(b), in which the domino edges are solid (domino edges ending at p_0 are not shown) and blocking edges are dashed arrows. Let L_q^α be a q-avoiding labeling with label length of α. We will define a subgraph G_q^α of $\mathcal{G}$ with minimum number of vertices and edges, and show that L_q^α exists if and only if there exists a valid (to be defined) G_q^α. We need the following definitions:

Definition 1 $(p_i, p_j) \in$ *DominoEdges is a* domino-reachable *edge from any p_k in $\mathcal{G}$, if there is a directed simple path $\pi : p_k \rightsquigarrow p_i$ of domino edges.*

Definition 2 $(p_i, p_j) \in$ *DominoEdges is an α-*terminating *edge from p_k, if (p_i, p_j) is a domino-reachable edge from p_k with a path π, where for each edge $e \in \pi$, $w(e) < \alpha$ and $w(p_i, p_j) \geq \alpha$. The path $p_k \rightsquigarrow p_j$ is also defined as an* α-terminating *path.*

Definition 3 $(p_i, p_j) \in$ *DominoEdges is an α-*critical *edge from p_k, if (p_i, p_j) is an α-terminating edge and $w(p_i, p_j) = \alpha$.*

Let ℓ_q be the label that contains the query point q. p_q is the corresponding point of ℓ_q. We define $G_q^\alpha = (P_q^\alpha, E_q^\alpha)$ as a subgraph of $\mathcal{G}$ containing all α-terminating paths from p_q. That is, P_q^α and E_q^α are the set of all vertices and edges on all α-terminating paths from p_q respectively (See Fig. 2). It is obvious that G_q^α is unique. Moreover, it is easy to see the following property.

Lemma 1 *If $\alpha \leq \beta$ then $G_q^\alpha \subseteq G_q^\beta$.*

Definition 4 *The internal nodes (not including the zero out-degree vertices) of G_q^α is denoted as I_q^α. Besides, the boundary edges (all edges that ends in a zero out-degree vertex) of G_q^α is denoted as B_q^α*

We are only concerned with the valid G_q^α to be defined below.

Definition 5 *G_q^α is valid if for all $p_i, p_j \in I_q^\alpha$ and $(p_i, p_j) \in$ BlockingEdges, we have $w(p_i, p_j) \geq \alpha$.*

2.1 Properties of the Optimal Solution

The following is the main property of the optimal solution.

Theorem 2 *There exists an L_q^α if and only if there exists a valid G_q^α.*

The proof is given in the following lemmas:

Lemma 3 *An L_q^α can be constructed from a valid G_q^α. Moreover, the minimum label length in L_q^α is α if at least one of the following conditions holds:*

1. *There is an α-critical edge in G_q^α.*
2. *There is a blocking edge with both ends in I_q^α and with weight of α.*

Proof. The operations required to construct L_q^α comes from the following steps:

1. Flip label ℓ_i for each domino edge $(p_i, p_j) \in E_q^\alpha$.
2. Resize one or both labels ℓ_i and ℓ_j to length $w(p_i, p_j)$ for each boundary edge $(p_i, p_j) \in B_q^\alpha$.
3. Resize one or both labels ℓ_i and ℓ_j to length $w(p_i, p_j)$ for each blocking edge (p_i, p_j) with both ends in I_q^α.

The generated labels are all larger than α since there is no resize operation to a length less than α. Moreover, no two labels may intersect since, otherwise, there must be a (domino or blocking) edge with weight less that α in $B_q^\alpha \cup I_q^\alpha$ which contradicts the validity of G_q^α. It is easy to verify that, if there is an α-critical edge or a blocking edge with both ends in I_q^α with weight α, then a label with length equal to α is generated. □

Lemma 4 *For any L_q^α, there exists a valid G_q^α.*

Proof. Define V as the set of points with flipped or resized labels, and $E = \{(p_i, p_j) | p_i, p_j \in V, (p_i, p_j) \in$ *DominoEdges*$\}$. Suppose π is an α-terminating path starting from q. This path should be in E since, otherwise, either there is a sequence of flips along this path not ending to a label with length at least α, or π is not an α-terminating path. So, E_q^α, which is the union of all α-terminating paths, is a subset of E, and hence $V_q^\alpha \subseteq V$. Since the initial labeling is valid, there is no blocking edge with both ends in internal nodes of V, and hence there is no blocking edge with weight less that α in V_q^α. So, a valid G_q^α exists. □

From Theorem 2, we can conclude that it is sufficient to check the existence of valid G_q^α. Lemma 5 proves that we only need to check this for at most $O(n)$ values of α's, and this can be searched more effectively from the fact given in Lemma 6.

Lemma 5 *The optimal label length belongs to the set $\{w(e) | e \in \mathcal{E}\}$, which has at most $O(n)$ elements.*

Proof. For the first part, assume that the optimal label length α does not belong to $\{w(e) | e \in E\}$. So, there should be a label of length α, say ℓ_i, in the optimal labeling. This label is resized to α to resolve an intersection with some other label, say ℓ_j. Obviously, the best resizing for these labels is $w(p_i, p_j) > \alpha$ and hence the value of α is not optimal.

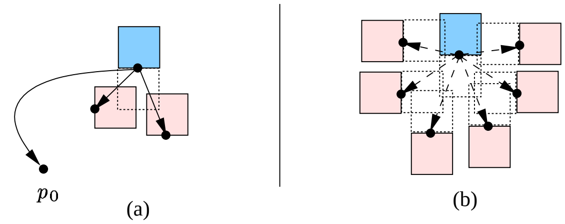

Figure 3: Maximum number of domino and blocking edges from a given label (shown as darker label) in (a) and (b) respectively

For the second part, we count the number of edges in the conflict graph $\mathcal{G}$. It is obvious that for each $p_i \in \mathcal{P}$, there is at most three edges in *DominoEdges* (Fig. 3(a)) and at most six edges in *BlockingEdges* (Fig. 3(b)) starting at p_i. Therefore, $|E| \leq 9n$ hence the set $\{w(e) | e \in E\}$ has $O(n)$ members. □

It is easy to see the following.

Lemma 6 *If there is no valid G_q^α then there is no valid G_q^β for all $\beta > \alpha$.*

3 The Optimal Algorithm

For each label, we define a region inside it, denoted by *flipping region*, to shows that the corresponding label must be flipped only if the moving point is inside that region. This region is a simple polygon with constant number of edges. Later in this section, we will briefly describe how this region is calculated.

The key routine in the optimal algorithm is "Finding G_q^α" that generates the subgraph G_q^α with the maximum value of α for any given label ℓ_q. This routine has two purposes: in the preprocessing phase it is used to find the flipping region, and in the online part of the algorithm we extract the required flip and resize operations from the output subgraph. We will show that this routine finds G_q^α in $O(k)$ amortized time where k is the size of the subgraph(or equally the size of the optimal output).

3.1 The Preprocessing Phase

This phase consists of the following steps:

1. Build the conflict graph $\mathcal{G}$,
2. Calculate the weight function $w(e)$ for all edges,
3. Create a sorted list of all weight of $\alpha_i = w(e)$ and assume that $\alpha_1 \leq \alpha_2 \leq \ldots < 1$,
4. Construct a point location data structure on the initial labels [8],
5. Calculate and store flipping region for each label ℓ_i.

Lemma 7 *The preprocessing phase needs $O(n^2)$ time.*

Proof. The conflict graph can be built using a simple vertical sweep line algorithm keeping track of all intersecting labels with the sweep line in $O(n \lg n)$ time. The value of $w(e)$ can also be computed in $O(1)$ time per each edge. Steps 3 and 4 also take $O(n \lg n)$ time. To calculate the flipping region, which has constant number of edges, in step 5, we need to build $G_q^{\alpha_i}$ that needs $O(n)$ time for each label. So, the overall time required in the preprocessing phase is $O(n^2)$. □

3.2 Finding G_q^α

Given a label ℓ_q, the routine starts with building the subgraph $G_q^{\alpha_1}$. Hereafter, the routine builds $G_q^{\alpha_{i+1}}$ by adding some vertices and edges (possibly empty) to $G_q^{\alpha_i}$. The following properties are used in constructing $G_q^{\alpha_{i+1}}$:

1. $G_q^{\alpha_1} \subseteq G_q^{\alpha_2} \subseteq G_q^{\alpha_3} \subseteq \ldots$ (definition of $G_q^{\alpha_i}$),

2. All edges of $G_q^{\alpha_{i+1}}$ connected to a leaf vertex of $G_q^{\alpha_{i+1}}$ have weights at least α_i (Def. 2),

3. The value $\beta_{i+1} = \min(\{w(u,v) | (u,v) \in I_q^{\alpha_{i+1}}\})$ is an upper bound of the optimal label length (Def. 5),

4. $G_q^{\alpha_{i+1}}$ is valid if and only if $\beta_{i+1} \geq \alpha_{i+1}$ (Def. 5), and

5. Finally, the recursive definition of $G_q^{\alpha_{i+1}}$ (definition of $G_q^{\alpha_{i+1}}$ and Def. 5) is as follows:

$$G_q^{\alpha_{i+1}} = G_q^{\alpha_i} \cup \{G_v^{\alpha_{i+1}} | (u,v) \in B_q^{\alpha_i}, w(u,v) = \alpha_i\}.$$

Using the above recursive definition of $G_q^{\alpha_{i+1}}$ a simple incremental algorithm can be proposed:

Finding G_q^α

Input: Conflict graph $\mathcal{G}$, and a label ℓ_q,
Output: The subgraph G_q^α with the maximum value of α
Algorithm:

1. Let $i = 1$ and construct $G_q^{\alpha_1}$.
2. **while true do**
   (a) Compute $\{G_v^{\alpha_{i+1}} | (u,v) \in B_q^{\alpha_i}, w(u,v) = \alpha_i\}$ and build $G_q^{\alpha_{i+1}}$.
   (b) Compute sets $B_q^{\alpha_{i+1}}$, $I_q^{\alpha_{i+1}}$, and β_{i+1}.
   (c) **if** $\beta_{i+1} < \alpha_{i+1}$ **then**
      i. return $G_q^{\alpha_i}$ and **terminate**.
   (d) Let $i = i + 1$.
3. **end while**

Lemma 8 *The $G_q^{\alpha_i}$ for all values of α_i can be constructed in ascending order of α_i in overall $O(n)$ time.*

Proof. Any edge in the conflict graph is visited at most a constant number of times: A domino edge is visited at most once, and a blocking edge is visited at most twice (a blocking edge is checked whenever one of its ends is added to the internal vertex set). With appropriate data structures for maintaining sets I and B, each visit to an edge can be implemented in $O(1)$. □

Having $G_q^{\alpha_i}$ for an ℓ_q, we can construct the flipping region of ℓ_q as follows. Shrink all flipped labels ℓ_j (according to $G_q^{\alpha_i}$) that have intersection with original location of ℓ_q to size α, and then obtain the flipping region of ℓ_q, which is a polygon, from the intersection of ℓ_q with those shrunk labels. It is easy to see that the flipping region for each label has a constant number of edges, hence deciding to flip the label containing the query point can be done in $O(1)$ time.

It is obvious that if the moving point goes into ℓ_q but not in the flipping region of ℓ_q, the $G_q^{\alpha_i}$ will generate a labeling with size less than α_i. So, the optimal solution is to resize ℓ_q instead of flipping it.

3.3 The Optimal Label Updating Algorithm

Using the "Finding G_q^α", the label updating algorithm can be as simple as follows:

The Optimal Label Updating Algorithm

For each position q of the moving point do the following steps:

1. Locate the label ℓ_q containing q.
2. **if** no such label exists **then** $\mathcal{L}$ is the optimal labeling and **terminate**.
3. **if** q is not in the flipping region of ℓ_q, **then** resize ℓ_q to obtain optimal labeling,
4. Call "Finding G_q^α" and write the required operations according to G_q^α.

The above algorithm along with Lemma 8 yields the following theorem:

Theorem 9 *Given a moving point q on a labeling L, the time required to generate an updated q-avoiding labeling is $O(\lg n + k)$ where k is the number of operations required to update L.*

4 Conclusions

In this paper, we introduced the problem of updating a squared axis-parallel labeled map to avoid a moving point. We modeled the initial labeling and update operations with a directed multi-graph with at most $O(n)$ edges and vertices called conflict graph. We also showed that given a point q, the optimal q-avoiding labeling corresponds to a subgraph of the conflict graph that can be found in time $O(\lg n + k)$ where k is the number of update operations.

If the path of the moving object is known initially, say a straight line, we cannot gain any performance with the technique used here. Knowing where the moving object will go, only let us foretell the next event (when a label updating action is required to update the map).

The proposed data structure in this paper only supports one moving point, but it can simply be extended to a constant number of moving points that gives the optimal labeling with the same time bounds.

The technique used in this paper can be extended to other labeling schemes, like axis parallel rectangular labels. The only difference is that the conflict graph may have $O(n^2)$ edges. The algorithm still produces the optimal labeling in $O(\lg n + k)$ time where k is $O(n^2)$.

References

[1] Marks, J., Shieber, S.: The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard CS (1991)

[2] Duncan, R., Qian, J., Vigneron, A., Zhu, B.: Polynomial time algorithms for three-label point labeling. Theoretical Computer Science **296** (2003) 75–87

[3] Qin, Z., Wolff, A., Xu, Y., Zhu, B.: New algorithms for two-label point labeling. In Paterson, M., ed.: Proc. 8th Annu. Europ. Symp. on Algorithms (ESA'00). Volume 1879 of Lecture Notes in Computer Science., Saarbrücken, Springer-Verlag (2000) 368–379

[4] Spriggs, M.J., Keil, J.M.: A new bound for map labeling with uniform circle pairs. Information Processing Letters **81** (2002) 47–53

[5] Strijk, T., Wolff, A.: Labeling points with circles. International Journal of Computational Geometry and Applications **11** (2001) 181–195

[6] Wolff, A., Thon, M., Xu, Y.: A simple factor-2/3 approximation algorithm for two-circle point labeling. International Journal of Computational Geometry and Applications **12** (2002) 269–281

[7] Doddi, S., Marathe, M.V., Mirzaian, A., Moret, B.M., Zhu, B.: Map labeling and its generalizations. In: Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms (SODA'97), New Orleans, LA (1997) 148–157

[8] Sarnak, N., Tarjan, R.E.: Planar point location using persistent search trees. Commun. ACM **29** (1986) 669–679