An Efficient Algorithm for Label Updating in 2PM Model to Avoid a Moving Object

Farshad Rostamabadi* Mohammad Ghodsi†

Abstract

In this paper, we present a simple and fast algorithm for updating labels of a set of points in presence of a moving point-shaped object. The labels are assumed to be axis parallel, unit length, square shaped, each attached exclusively to a point on one of its horizontal (vertical) edges, denoted by 2PM model. The updated labeling should include all labels, avoid the moving point with largest possible label length. We allow flip and resize operations on labels for updating a labeling. The known algorithm for this problem, where labels may be attached to their corresponding points on the middle of any edges, the 4PM model, uses $O(n^2)$ preprocessing time and $O(n)$ space to update the labeling in $O(\lg n + k)$ time, where k is the number of update operations (Rostamabadi and Ghodsi, CCCG'04). In this paper, we present a simpler and more efficient algorithm that uses $O(n \lg n)$ time and $O(n)$ space for preprocessing with simplified data structures and updates the labeling with the same time bound.

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 [2]. There are different variations of point-labeling that are discussed in [1, 3, 5, 6, 7].

In this paper, we are interested in a simple and efficient algorithm to update labels in a point-labeling map. We assume that our map is composed of a number of points each labeled by a unit-length axis-parallel square label. It is assumed that the point of each label appears in the middle of only one of its horizontal (vertical) edges. Hereafter, we denote this labeling model by 2PM[1]. Besides, a point-shaped object moves on the labeling and triggers an event every time it changes its location. Labels are allowed to be flipped or resized to avoid the moving object. The goal is to efficiently generate the updated labeling with maximum label length in response to each event triggered by the moving object.

A more general version of this problem, where each point can appear at the middle of any of four edges of its label, is considered in [4]; we refer to this labeling model as 4PM. Although, the proposed algorithm also works for our problem, but we are interested in a simpler algorithm with more efficient data structures.

Authors of [4] present a weighted and directed graph, *the conflict graph*, to convey the effect of all possible flip and resize operations. Since the conflict graph may have complex structure, the preprocessing phase requires $O(n^2)$ time in 4PM model.

We use the same definition of flip and resize operations as in [4], but we simplify the conflict graph definition by using some properties of the 2PM model, hence the preprocessing time reduces to $O(n \lg n)$. Besides, the result of the preprocessing phase can be stored in a more simpler data structures. We also show that the given labeling can be updated in $O(\lg n + k)$ time in response to each event triggered by the moving object, where k is the number of update operations.

2 Definitions

The label updating problem is precisely defined as follows. We are given a valid labeling L composed of points $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ and unit-length axis-parallel square labels $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_n\}$, where ℓ_i is attached to p_i on the mid-point of one of its horizontal (vertical) edges. The moving point generates an event whenever it changes its current position to a new location q. The problem is to update L and obtain a new *q-avoiding* labeling L_q for all points in $\mathcal{P}$, such that the moving point q does not intersect with any label in L_q, while the new labels are as close as possible to their original length. A label ℓ_i can be flipped over the edge containing its corresponding

*School of Computer Science, Institute for Studies in Fundamental Sciences, Tehran, Iran. rostamabadi@ipm.ir

†Department of Computer Engineering, Sharif University of Technology, and IPM School of Computer Science, Tehran, Iran. ghodsi@.sharif.edu

[1] 2P is a known two-position label modeling, and we add an M to it to denote that the point should appear at the middle of one of its horizontal (or vertical) edges of the label.

point p_i and the flipped position of ℓ_i is denoted by $f(\ell_i)$. ℓ_i can also be resized to any length, $\alpha \leq 1$ as long as p_i remains on the mid-point of its edge. The new location of ℓ_i is denoted by $r(\ell_i, \alpha)$. Such a final re-labeling is denoted by *q-avoiding optimum labeling*.

We name the above labeling model as 2PM. To be more precise, we formally define the 2PM and 4PM square-labeling models as follows:

Definition 1 *In 2PM model, every label is attached exclusively to a point on the middle of one of its horizontal (vertical) edges.*

Definition 2 *In 4PM model, every label is attached exclusively to a point on the middle of one of its edges.*

In next section, we formally define the conflict graph.

2.1 The Conflict Graph

The conflict graph $\mathcal{G} = (V, E)$, where $\mathcal{V} = V^+ \cup V^-$, is a directed weighted graph encoding all possible flip and resize operations. The conflict graph is both vertex and edge weighted. Let v_i be the representing vertex for label ℓ_i attached to the point p_i. The set V^+ (V^-) contains all vertices v_i where p_i is on the top (bottom) edge of ℓ_i.

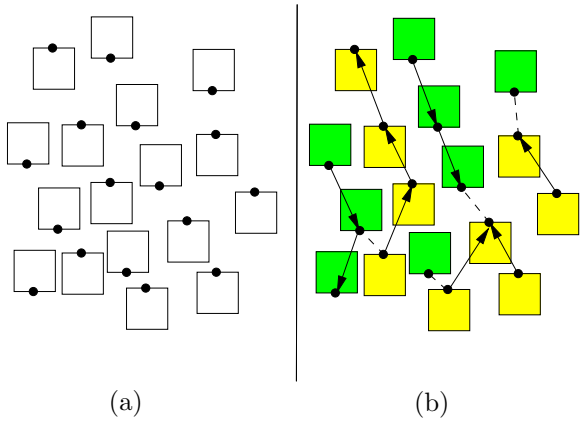
The edge set of the conflict graph, $\mathcal{E}$, is composed of three sets D^+, D^-, and B as follows. The edge set D^+ (D^-) contains all directed edges (v_i, v_j) where both v_i and v_j belongs to V^+ (V^-) and $f(\ell_i)$ intersects ℓ_j. If $f(\ell_i)$ intersects ℓ_j but v_i and v_j are in different vertex sets, then the undirected edge (v_i, v_j) belongs to B. More formally:

$\mathcal{E} = D^+ \cup D^- \cup B$, where
$D^+ = \{(v_i, v_j) | v_i, v_j \in V^+\}$, and
$D^- = \{(v_i, v_j) | v_i, v_j \in V^-\}$, and
$B = \{(v_i, v_j) | v_i$ and v_j are not in the same vertex set$\}$.

A sample input labeling and the generated conflict graph are shown in Figure 1. The D^+ and D^- edges are solid while the B edges are dashed. Besides, the vertices of V^+ have yellow (light gray) labels and the vertices of V^- have green (dark gray) labels. We define two induced subgraphs over the V^+ and V^- vertices in the following and will prove some important properties for them in the next section.

$$G^+ = (V^+, D^+),$$
$$G^- = (V^-, D^-).$$

In $\mathcal{G}$, an edge weight represents the optimal label length of a single flip (and possibly one resize) operation on a single label, and a vertex weight represents the optimal label length if multiple flip and resize operations are allowed over all labels in the map. To define the weight of an edge, we first introduce the g function below.



(a) (b)

Figure 1: (a) Initial labeled map. (b) The conflict graph: D^+ and D^- (solid) and B edges (dashed).

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 the $g(\ell_a, \ell_b)$.

Performing a flip operation on ℓ_i may cause an intersection between $f(\ell_i)$ and at most two other labels. Let ℓ_j be one of the intersecting labels with $f(\ell_i)$. We define the weight of the edge (v_i, v_j), for both directed and undirected edges equal to $g(f(\ell_i), \ell_j)$.

The weight assigned to v_i, which denoted by $w(v_i)$, represents the label length of a labeling when ℓ_i is assumed to be flipped and other labels may be flipped or resized to generate the maximum possible label length. Obviously, the weight of vertex v_i has a very close relation to the optimal labeling when the moving point is inside ℓ_i. So, we first prove some basic properties of the optimal labeling and then complete the definition of vertex weight.

3 Properties of The Optimal Solution

Let the moving point be at position q inside label ℓ_i, and L_i^α be the optimal q-avoiding labeling with minimum number of flip and resize operations. Besides, assume that all labels in L_i^α have length larger than $\alpha \leq 1$. We define $G_i^\alpha = (V_i^\alpha, E_i^\alpha)$ as a subgraph of $\mathcal{G}$ where V_i^α contains all vertices that their corresponding labels are flipped. Let E_i^α be the induced edges over V_i^α (all edges of $\mathcal{E}$ with both ends in V_i^α). The following lemmas show some basic properties of G_i^α.

Lemma 1 *If $v_i \in V^+$ then all vertices of V_i^α are in V^+.*

Proof. Assume that $f(\ell_i)$ intersects a label ℓ_j where $v_j \in V^-$. Obviously, flipping ℓ_j produces smaller labels of length $g(f(\ell_i), f(\ell_j))$ than $g(f(\ell_i), \ell_j)$. So, in the optimal solution, the vertex $v_j \in V^-$ can not belong to L_i^α. □

From the above lemma, we can conclude that if $v_i \in V^+$ then $G_i^\alpha \subseteq G^+$. Besides, by the symmetry of G^+ and G^-, we can also conclude that if $v_i \in V^-$ then $G_i^\alpha \subseteq G^-$

Lemma 2 *G_i^α is a DAG, rooted at v_i.*

Proof. Since all vertices of G_i^α belong to either V^+ or V^-, then all edges are upward or downward. So, G_i^α can not contain a loop, and is a DAG. Besides, it is easy to see that there is a directed path (a sequence of flips) from v_i to any other vertex in G_i^α hence v_i is the root vertex of G_i^α. □

Lemma 3 *Let v_j be a leaf vertex in G_i^α and (v_j, v_k) be an edge attached to v_j. Then one of the followings is true:*

1. *If (v_j, v_k) exists then $w(v_j, v_k) \geq \alpha$,*
2. *v_j has zero out-degree and (v_j, v_k) does not exist.*

Proof. Will appear in final version. □

4 Vertex weight definition

In this section, we will provide a bottom-up recursive definition and algorithm for vertex weights using lemmas from previous section. The weight of v_i is the minimum label length in the optimal updated labeling when ℓ_i is flipped (equivalently, the query point is inside ℓ_i) and also corresponds to a subgraph of G^+ or G^- generating the optimal updated labeling. For simplicity, we focus on G^+, but all the definitions and the algorithm can also be applied to G^-.

First, we define the weight of zero out-degree vertices of G^+. Consider $v_i \in G^+$ where the out-degree of vertex v_i in G^+, denoted by $d^+_{\text{out}}(v_i)$, is zero. If v_i is also a zero out-degree vertex in $\mathcal{G}$, then $f(\ell_i)$ has no intersection with other labels hence the optimal updated labeling after flipping ℓ_i has no label of length less than one. Otherwise, the weight of undirected edges starting from v_i will define the vertex weight. More precisely, where $d^+_{\text{out}}(v_i) = 0$, we have:

$$w(v_i) = \min(\{1\} \cup \{w(v_i, v_j) | (v_i, v_j) \in B\}).$$

Second, for non-zero out-degree vertices, consider an edge $(v_i, v_j) \in D^+$ and assume that ℓ_i is flipped. There are two alternatives to remove the intersection between $f(\ell_i)$ and ℓ_j: (1) Resize both intersecting labels to some smaller length, and (2) Flip ℓ_j and solve the problem recursively (if applicable). The minimum

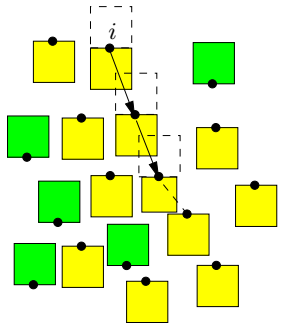


Figure 2: G_i^α and the optimal generated L_i^α.

generated label length is $g(f(\ell_i), \ell_j)$ in the former case (according to the definition of edge weight) and $w(v_j)$ in the latter case. We summarise the above description into a function, denoted by h, in the following:

$$h(v_i, v_j) = \begin{cases} \max(w(v_i, v_j), w(v_j)) & , v_j \in V^+ \\ w(v_i, v_j) & , v_j \notin V^+ \end{cases}$$

Using above function, the vertex weights can precisely be defined as:

$$w(v_i) = \begin{cases} \min\{\{1\} \cup \{w(v_i, v_j) | (v_i, v_j) \in B\}\} \\ \qquad\qquad , d^+_{\text{out}}(v_i) = 0 \\ \min\{h(v_i, v_j) | (v_i, v_j) \in \mathcal{E}\} \\ \qquad\qquad , d^+_{\text{out}}(v_i) \neq 0 \end{cases}$$

Obviously, using a simple bottom-up algorithm, all vertex weights can be calculated in $O(n)$ time.

The value of $w(v_i)$ defines a DAG subgraph, denoted by G_i, rooted at v_i and corresponds to the optimal labeling obtained by flipping ℓ_i. G_i is generated implicitly in the bottom-up calculation of $w(v_i)$ in time $O(n)$ for all vertices. But, if the value of $w(v_i)$ is available, it can simply be obtained using a top-down algorithm in time $O(|G_i|)$.

In figure 2 an optimal labeling L_i^α along with the corresponding G_i is shown for an arbitrary vertex.

Assume that the query point is inside ℓ_i and we have to flip ℓ_i. Having $w(v_i)$ been calculated in the preprocessing phase, we build G_i and produce the required update operations, in time $k = O(|G_i|)$, using the following transformation:

1. Flip label ℓ_m iff v_m is not a leaf vertex in G_i,
2. Flip and resize ℓ_j to $\min\{w(v_j, v_k) | (v_j, v_k) \notin G_i\}$ for all leaf vertices v_j.
3. Resize ℓ_k to length $\min\{w(v_j, v_k) | \ell_k \cap \ell_j \neq \emptyset \wedge v_j \in G_i \wedge (v_j, v_k) \notin G_i\}$ for all $v_k \notin G_i$.

According to lemma 3, it is easy to see the following theorem.

Theorem 4 *The vertex subgraph G_i of vertex v_i corresponds to the optimal labeling with labels of length*

at least $w(v_i)$, when the query point is inside ℓ_i and ℓ_i has to be flipped.

5 The Optimal Algorithm

In this section, we provide the optimal algorithm for label updating in 2PM model. The algorithm has a preprocessing and an online phase. In the former phase, the optimal labeling lengths are computed and stored for each label (vertex) in overall time $O(n \lg n)$. In the later phase, we compute the operations required to update the labeling using the optimal labeling lengths in time $O(\lg n + k)$ where k is the number of flip and resize operations.

In the preprocessing phase the following steps should be taken:

Preprocessing Phase

1. Build the conflict graph and compute edge and vertices weights.
2. Build a point location data structure on all labels.

In the online phase, for a given point q, the optimal labeling can be generated with the following steps:

Online Phase

1. Locate ℓ_i containing q.
2. **if** no such label was found, **then** the original labeling is optimal and **exit**.
3. Maximize γ where $r(\ell_i, \gamma)$ does not contain q.
4. **if** $\gamma \geq w(v_i)$ **then** the resize operation $r(\ell_i, \gamma)$ generate the optimal labeling and **exit**.
5. Calculate G_i.
6. Transform G_i to the optimal labeling.

It is easy to see the following theorem.

Theorem 5 *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 operation required to update L.*

6 Conclusion

In this paper, we introduced the problem of updating a squared axis-parallel labeled map to avoid a moving point. We develop a simple and fast algorithm and improve the previous results from $O(n^2)$ preprocessing time to $O(n \lg n)$ with the same space and query time. We modeled the initial labeling and update operations with a weighted 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.

References

[1] Rob Duncan, Jianbo Qian, Antoine Vigneron, and Binhai Zhu. Polynomial time algorithms for three-label point labeling. *Theoretical Computer Science*, 296(1):75–87, 2003.

[2] Joe Marks and Stuart Shieber. The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard CS, 1991.

[3] Zhongping Qin, Alexander Wolff, Yinfeng Xu, and Binhai Zhu. New algorithms for two-label point labeling. In Mike Paterson, editor, *Proc. 8th Annu. Europ. Symp. on Algorithms (ESA'00)*, volume 1879 of *Lecture Notes in Computer Science*, pages 368–379, Saarbrücken, 5–8 September 2000. Springer-Verlag.

[4] Farshad Rostamabadi and Mohammad Ghodsi. A fast algorithm for updating a labeling to avoid a moving point. In *Proceedings of the 16th Canadian Conference on Computational Geometry (CCCG'04)*, pages 204–208, 2004.

[5] Michael J. Spriggs and J. Mark Keil. A new bound for map labeling with uniform circle pairs. *Information Processing Letters*, 81(1):47–53, 2002.

[6] Tycho Strijk and Alexander Wolff. Labeling points with circles. *International Journal of Computational Geometry and Applications*, 11(2):181–195, April 2001.

[7] Alexander Wolff, Michael Thon, and Yinfeng Xu. A simple factor-2/3 approximation algorithm for two-circle point labeling. *International Journal of Computational Geometry and Applications*, 12(4):269–281, 2002.