Dynamic point labeling is strongly PSPACE-hard

Dynamic point labeling is strongly PSPACE-hard

Kevin Buchin* Dirk H.P. Gerrits*

Abstract

An important but strongly NP-hard problem in automated cartography is how to best place textual labels for point features on a static map. We examine the complexity of various generalizations of this problem for dynamic and/or interactive maps. Specifically, we show that it is strongly PSPACE-hard to obtain a smooth dynamic labeling (function from time to static labelings) when the points move, when points are added and removed, or when the user pans, rotates, and/or zooms their view of the points.

1 Introduction

Map labeling involves associating textual *labels* with certain *features* on a map such as cities (points), roads (polylines), and lakes (polygons). This task takes considerable time to do manually, and for some applications *cannot* be done manually beforehand. In air traffic control, for example, a set of moving points (airplanes) has to be labeled at all times. In interactive maps users may pan, rotate, and/or zoom their view of the map, which may require relabeling.

Static point labeling. A good labeling for a point set has legible labels, and an unambiguous association between the labels and the points. This has been formalized by regarding the labels as axis-aligned rectangles slightly larger than the text they contain, which must be placed without overlap so that each contains the point it labels on its boundary. Not all placements are equally desirable, and as such various *label models* have been proposed, which specify the subset of allowed positions for the labels. In the *fixed-position models*, every point has a finite number of label candidates. In particular, in the 1-, 2-, and 4-position models a subset of 1, 2, or 4 corners is designated (the same subset for all labels) and each label must have one of these corners coincide with the point it labels. The *slider models* generalize this. In the 1-slider models one side of each label is designated, but the label may contain its point anywhere on this side. In the 2-slider models there is a choice between two opposite sides of the label, and in the 4-slider model the label can have the point anywhere on its boundary.

Ideally, one would like to label all points with pairwise non-intersecting labels. This is not always possible, however, and the decision problem is strongly NP-complete for the 4-position [2] and 4-slider [5] models, even if all labels are unit squares. We may deal with this difficulty in several ways. Firstly, we may reduce the font size, that is, shrink labels. The *size-maximization problem* asks to label all points with pairwise non-intersecting labels of maximal size. Secondly, we may remove labels. The *(weighted) number-maximization problem* asks to label a maximum-cardinality (maximum-weight) subset of the points with pairwise non-intersecting labels of given dimensions. Thirdly, we may allow labels to overlap, but try to keep such occurrences to a minimum. The *free-label-maximization problem* asks for all points to be labeled with labels of given dimensions, maximizing the number of non-intersecting labels. All of these problems are strongly NP-hard for the 4-position and 4-slider models because of the above result.

Dynamic point labeling. A natural generalization of static point labeling is dynamic point labeling. Here the point set P changes over time, by points being added and removed, and/or by points moving continuously. We then seek a *dynamic labeling* $\mathcal{L}$, which for all t assigns a static labeling $\mathcal{L}(t)$ to the static points $P(t)$ present at time t. For the 4-slider model we require that $\mathcal{L}$ is continuous in the sense that, if a point p has a label over a non-empty time interval, then the label must move continuously over that interval. For the fixed-position and 2-slider models we must allow labels to make "jumps". We only allow p's label to jump from position A to position B, however, if there is no candidate position C in between A and B in clockwise (or counter-clockwise) order around p. Thus, we allow horizontal and vertical (but no diagonal) jumps for the 4-position model. For a 2-slider model we only allow jumps from an endpoint of one slider to the "same" endpoint of the other slider.

For static point labeling, both practical heuristic algorithms and theoretical algorithms with guaranteed approximation ratios abound. Dynamic point labeling, however, has seen very few theoretical results. Been et al. [1] studied unweighted number-maximization for points under continuous zooming, giving constant-factor approximations for unit-square labels in the 1-position model. Gemsa et al. [3] similarly studied continuous rotation, and give a PTAS for unit-square labels in the 1-position model. Treatment of other label models and more general point trajectories are sorely missing.

*Dept. of Computer Science, TU Eindhoven, the Netherlands, k.a.buchin@tue.nl, dirk@dirkgerrits.com

Our results. We believe the relative lack of theoretical results for dynamic point labeling is not due to a lack of attempts. Intuitively, dynamic point labeling should be much harder than its static counterpart. We prove and quantify this intuition. Specifically, we prove for unit-square labels in the 4-position, 2-slider, and 4-slider models that deciding whether there exists a dynamic labeling without intersections is strongly PSPACE-complete. This is the case when points are added or removed from the point set, when (some of) the points move, and when the points set is panned, rotated, or zoomed within a finite viewport. Any dynamic generalization of the mentioned static optimization problems is therefore strongly PSPACE-hard in these settings. Additionally, we prove that label-size maximization on dynamic point sets admits no PTAS unless P=PSPACE. To save space in this extended abstract, we present our proofs in picture form, with minimal accompanying text.

2 Non-deterministic constraint logic

To prove PSPACE-hardness of dynamic point labeling, we will reduce from *non-deterministic constraint logic* (NCL) [4], which is a sort of abstract, single-player game. The game board is a *constraint graph*: an undirected graph with weights on both the vertices and the edges. A *configuration* of the constraint graph specifies an orientation for each of its edges. A configuration is *legal* if and only if each vertex v's *inflow* (the summed weight of its incoming edges) is at least v's own weight. To make a *move* is to reverse a single edge in a legal configuration such that the resulting configuration is again legal. For this game each of the following questions is PSPACE-complete [4].

- *Configuration-to-configuration NCL*: Given two legal configurations A and B, is there a sequence of moves transforming A into B?
- *Configuration-to-edge NCL*: Given a legal configuration A and an orientation for a single edge e_B, is there a sequence of moves transforming A into a legal configuration B in which e_B has the specified orientation?
- *Edge-to-edge NCL*: Given orientations for edges e_A and e_B, do there exist legal configurations A and B, and a sequence of moves transforming the one into the other, such that e_A has the specified orientation in A and e_B has the specified orientation in B?

These decision problems remain PSPACE-complete even for planar, 3-regular constraint graphs consisting only of AND vertices and protected OR vertices. An *AND vertex* has a weight of 2, and its three incident edges have weights 1, 1, 2. To orient the weight-2 edge away from the vertex requires both weight-1 edges to be oriented towards the vertex. An *OR vertex* and its three incident edges all have weight 2. Thus at least one edge needs to be oriented towards the vertex at all times. An OR vertex is called *protected* if it has two edges that, because of constraints imposed on them by the rest of the constraint graph, cannot both be directed inward.

Gadgets. Figure 1 shows the *gadgets* we will employ in our reduction. We call the points with dark gray labels *blockers*, as we can consider these labels fixed obstacles (labeling them differently than shown will only restrict the placement of other labels further). In the 4-position model, the depicted blockers restrict the light gray labels marked A, B, and C to two possible positions each. We call the label position closest to the center of the vertex gadget *inward*, and the other *outward*. In the figure, labels A and B are placed inward, and label C is placed outward. In the slider models, labels can take on any position in between these two extremes, but we may assume that only a very small range of positions is actually used. Consider, for example, label A. Without moving A', we can only move A up by at most ε, or left by at most 2ε. We refer to this range of positions as *inward*, and define it similarly for B and C. If (and only if) A' is moved left by at least $1-\varepsilon$, then A can move further upward. We may then move A all the way to its uppermost position, and there is no reason not to do so. Thus we may define *outward* as in the 4-position model. With this terminology in place, we shall prove that our gadgets faithfully simulate a constraint graph, with labels placed *inward* corresponding to edges directed *out* of a vertex, and labels placed *outward* corresponding to edges directed *into* a vertex.

Theorem 1 *Let G be a planar constraint graph with n vertices, and let ε and δ be two real numbers with $0 < \varepsilon \leqslant \delta < 1 - 3\varepsilon$. One can then construct a point set $P = P(G, \delta, \varepsilon)$ with the following properties:*

- *the size of P, and the coordinates of all points in P, are polynomially bounded in n,*
- *for any $s \in [1, 1 + \delta - \varepsilon]$, there is a one-to-one correspondence between legal configurations of G and overlap-free static labelings of P with $s \times s$ square labels in the 4-position model, and*
- *there is a sequence of moves transforming one legal configuration of G into another if and only if there is a dynamic labeling transforming the corresponding static labelings of P into each other.*

When $\delta < 1/3 - 4\varepsilon/3$, the same results hold for the 2- and 4-slider models.

Proof. Because of the structure of Hearn and Demaine's hardness proof of NCL [4], we may assume that the given constraint graph G is embedded on a grid with its vertices on grid vertices and its edges being interior-disjoint paths along grid lines. We turn this grid by 45° relative to the coordinate axes, and replace the vertices and edges by appropriate gadgets.

Figure 1(a) shows an OR gadget that works if we assume that A and B are never both placed outward.

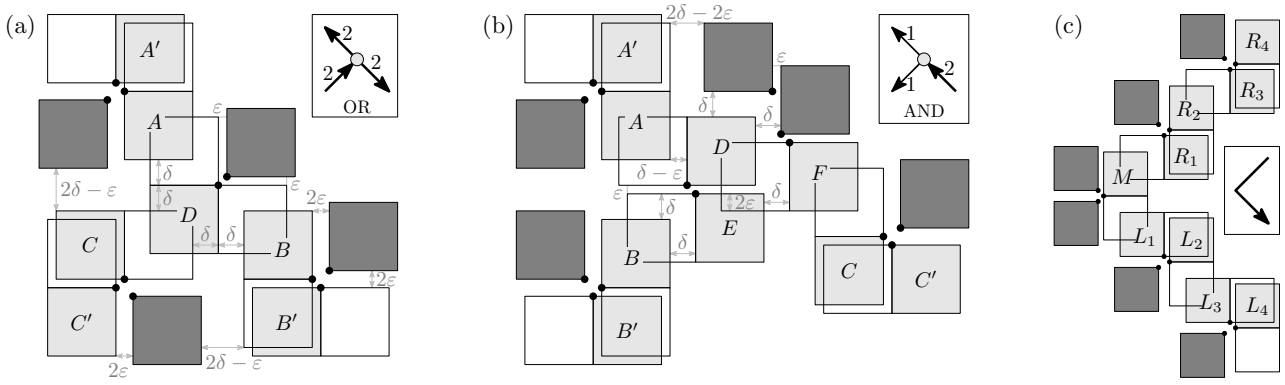

Figure 1: Gadgets simulating NCL's (a) protected OR vertices, (b) AND vertices, and (c) edges. The insets in the top-right corners show the NCL constructions being simulated.

This means it simulates a protected OR vertex where edges A and B are never both directed into the vertex. Figure 1(b) shows an AND gadget, where A and B represent the weight-1 edges. Figure 1(c) shows an edge gadget to connect vertex gadgets over longer distances. It is possible for both ends of an edge gadget to be placed inward for their respective vertex gadgets, however, this is of no concern as this corresponds to an edge of the constraint graph being directed into neither of its two incident vertices. If all inflow constraints are satisfied with such edges present, then orienting them arbitrarily does not disturb this.

If we scale the grid appropriately, then the gadgets will fit together so as to form the desired point set P. We omit the proof of P's claimed properties; it can be reconstructed by ruminating on Figure 1. □

3 Hardness of dynamic point labeling

In this section we will define a number of dynamic point-labeling problems in an offline setting. That is, we assume we are given the arrival and departure times of all points, as well as their trajectories and the "trajectory" of the viewport. We will seek a dynamic labeling $\mathcal{L}$ for the time interval $[a, b]$, and distinguish four cases based on whether or not the static labelings $\mathcal{L}(a)$ and $\mathcal{L}(b)$ are pre-specified. We will show PSPACE-hardness by reduction from configuration-to-edge NCL for the case where $\mathcal{L}(a)$ is given, and omit the symmetric case where $\mathcal{L}(b)$ is given instead. We also omit the cases where neither or both static labelings are given, which can be proven by similar reductions from edge-to-edge and configuration-to-configuration NCL, respectively. (The sole exception is zooming, where pre-specifying neither static labeling makes the problem "merely" NP-complete.)

Theorem 2 *The following decision problem is strongly PSPACE-complete for the 4-position, 2-slider, and 4-slider label models.*

Given: *A dynamic point set P (with given trajectories, arrival times, and departure times), numbers a and b with $a < b$, and a static labeling $\mathcal{L}(a)$ for $P(a)$.*

Decide: *Whether there exists a dynamic labeling $\mathcal{L}$ for P respecting $\mathcal{L}(a)$ that labels all points with non-overlapping unit-square labels over the time interval $[a, b]$.*

Additionally, unless P = PSPACE, the maximum label size for which there is such an $\mathcal{L}$ cannot be $(4/3 - \varepsilon')$-approximated in polynomial time for any $\varepsilon' > 0$. For the 4-position model, this holds even for a $(2 - \varepsilon')$-approximation.

All of the above remains true when

- *all points are stationary, and during $[a, b]$ one point is removed and one point is added,*
- *no points are added or removed, and all points move at the same, constant speed along straight-line trajectories, or*
- *no points are added or removed, and all points but one are stationary.*

Proof. We only prove the claim of PSPACE-hardness, which we do by reducing from configuration-to-edge NCL. Thus we are given a constraint graph G with a legal starting configuration A and the goal orientation of a single edge e_B, and want to decide whether there is a sequence of moves on G starting from A that will result in e_B having its specified orientation. By Theorem 1, we may construct a point set $P = P(G, \delta, \varepsilon)$ and a valid static labeling $\mathcal{L}(a)$ corresponding to configuration A. Now pick one blocker $q \in P$ in the edge gadget for e_B, and suppose $p \in P$ is the point for which q blocks some label candidate(s). We now make q move at a constant speed of $v = 2\varepsilon/(b - a)$ towards the nearest non-blocked candidate of p. Figure 2(a) shows the end result at time b: the edge gadget has become constrained to a single orientation around p. Thus there is a sequence of moves on A resulting in e_B having the specified orientation if and only if there is a dynamic labeling $\mathcal{L}$ for this dynamic point set starting at $\mathcal{L}(a)$. The same result may be obtained by moving q at speed $v/2$ and moving all other points at speed $v/2$ in the opposite direction. Alternatively, we may remove the point q from P and re-insert it at its modified location. □

In interactive mapping applications, users are presented with a rectangular *viewport* V showing a portion of a larger map. By dynamically panning, rotating, and/or zooming the map, the user controls which portion of the map is displayed at any given time. The task of labeling the points inside V can be seen as a special case of labeling dynamic point sets. Continuous panning, rotation, and zooming of the map may cause points to enter or leave V at its boundary, and causes all points within V to move on continuous trajectories. We will require only the points inside V to be labeled, and these labels must be fully contained in V. Points outside of V need not be labeled, but we may wish to do so in order to ensure a smooth dynamic labeling. Whether we do so or not makes no difference in the complexity of these problems.

Theorem 3 *The following decision problem is strongly PSPACE-complete for the 4-position, 2-slider, and 4-slider label models.*

Given: *A closed rectangle V, a points set P being panned, rotated, or zoomed in a single direction at constant speed, numbers a and b with $a < b$, and a static labeling $\mathcal{L}(a)$ for $P(a)$.*
Decide: *Whether there exists a dynamic labeling $\mathcal{L}$ for P respecting $\mathcal{L}(a)$ that labels $P(t) \cap V$ with non-overlapping unit-square labels inside V for all $t \in [a, b]$.*

Proof. We reduce from configuration-to-edge NCL in a way similar to Theorem 2, but we now make sure that G is laid out on the grid in such a way that e_B is on the far right. Next, we add an additional points q' to the right of e_B by $1 - 3\varepsilon$, as in Figure 2(b). We then construct a viewport rectangle V with its right side being ε to the left of q'. If we then start panning so that the points move to the left, q' will enter V after a distance ε. We must then label q' in such a way that e_B becomes constrained to a single orientation. The same result can be achieved by rotating the points clockwise, assuming the center of viewport is above q'. Unlike with panning, rotating also changes the distances between the points inside of the gadgets. But if the initial distance between q' and V is small enough, q' will enter V before the gadgets are disturbed. Zooming out similarly makes q' enter V. □

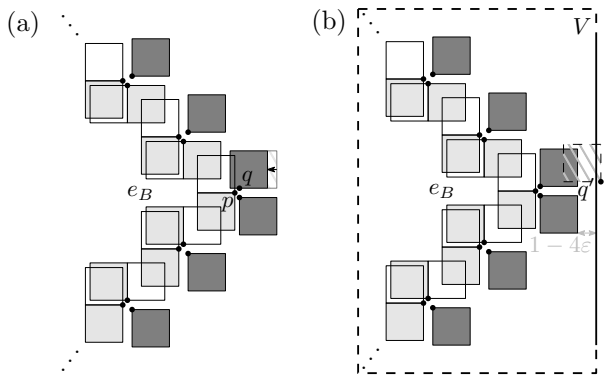

Figure 2: Reduction from configuration-to-edge NCL to dynamic labeling of (a) moving points, and (b) static points under panning of the viewport V.

4 Conclusion

We have examined the complexity of dynamic point labeling. For a set of points where points may be added or removed over time, and where the points present move along continuous trajectories, we seek a smooth function from time to static point labelings. For the 4-position, 2-slider, and 4-slider models we have shown that deciding whether there exists such a labeling in which no labels ever overlap is strongly PSPACE-complete. In addition, finding the maximum label size at which such a labeling does exist admits no PTAS unless P=PSPACE. For the 4-position model a 2-approximation is the best that can be hoped for, and for the slider models a 4/3-approximation. The PSPACE-completeness results also apply for special cases of dynamic point labeling in which the point set is panned, rotated, or zoomed inside a fixed viewport.

It remains to examine the complexity of other label models, specifically the 1- and 2-position models, as well as the 1-slider model. Presumably the decision problem is easier for these models, as in the case of static point labeling. It may also be of interest to re-examine the special cases of zooming and rotation with an infinite viewport. Our reduction currently uses points at the boundary of the viewport, but we believe this can be avoided. Most importantly, perhaps, is the continued pursuit of approximation algorithms for dynamic point labeling.

References

[1] K. Been, M. Nöllenburg, S.-H. Poon, and A. Wolff. Optimizing active ranges for consistent dynamic map labeling. *Comput. Geom. Theory Appl.*, 43(3):312–328, 2010.

[2] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 281–288, 1991.

[3] A. Gemsa, M. Nöllenburg, and I. Rutter. Consistent labeling of rotating maps. In F. Dehne, J. Iacono, and J.-R. Sack, editors, *Proc. 11th Internat. Sympos. Algorithms and Data Structures*, pages 451–462, 2009.

[4] R. A. Hearn and E. D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoret. Comput. Sci.*, 343(1–2):72–96, 2005.

[5] M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Comput. Geom. Theory Appl.*, 13:21–47, 1999.